Effort Estimation Methods in Software Development using Machine Learning Algorithms

Shashank Mouli Satapathy

Department of Computer Science and Engineering
National Institute of Technology Rourkela

Effort Estimation Methods in Software Development using Machine Learning Algorithms

Thesis submitted to the

National Institute of Technology Rourkela

in partial fulfillment of the requirements

of the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Shashank Mouli Satapathy

(Roll Number: 512CS104)

under the supervision of

Prof. Santanu Kumar Rath

October 2016

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Computer Science and Engineering
National Institute of Technology Rourkela

October 25, 2016

Certificate of Examination

Roll Number: 512CS104
Name: Shashank Mouli Satapathy
Title of Dissertation: Effort Estimation Methods in Software Development using Machine Learning Algorithms

We the below signed, after checking the dissertation mentioned above and the official record book (s) of the student, hereby state our approval of the dissertation submitted in partial fulfillment of the requirements of the degree of Doctor of Philosophy in Computer Science and Engineering at National Institute of Technology Rourkela. We are satisfied with the volume, quality, correctness, and originality of the work.

Santanu Kumar Rath
Principal Supervisor

Durga Prasad Mohapatra
Member (DSC)

Pabitra Mohan Khilar
Member (DSC)

Susmita Das
Member (DSC)

Swapan Bhattacharya
Examiner

Sarat Kumar Patra
Chairman (DSC)

Computer Science and Engineering
National Institute of Technology Rourkela

Prof. Santanu Kumar Rath
Professor

October 25, 2016

Supervisor's Certificate

This is to certify that the work presented in this dissertation entitled "*Effort Estimation Methods in Software Development using Machine Learning Algorithms*" by "*Shashank Mouli Satapathy*", Roll Number 512CS104, is a record of original research carried out by him/her under my supervision and guidance in partial fulfillment of the requirements of the degree of *Doctor of Philosophy* in *Computer Science and Engineering*. Neither this dissertation nor any part of it has been submitted for any degree or diploma to any institute or university in India or abroad.

Santanu Kumar Rath

This thesis is dedicated to my family.

For their endless love, support and encouragement

Declaration of Originality

I, Shashank Mouli Satapathy, Roll Number 512cs104 hereby declare that this dissertation entitled "*Effort Estimation Methods in Software Development using Machine Learning Algorithms*" represents my original work carried out as a doctoral student of NIT Rourkela and, to the best of my knowledge, it contains no material previously published or written by another person, nor any material presented for the award of any other degree or diploma of NIT Rourkela or any other institution. Any contribution made to this research by others, with whom I have worked at NIT Rourkela or elsewhere, is explicitly acknowledged in the dissertation. Works of other authors cited in this dissertation have been duly acknowledged under the section "Bibliography". I have also submitted my original research records to the scrutiny committee for evaluation of my dissertation.

I am fully aware that in case of any non-compliance detected in future, the Senate of NIT Rourkela may withdraw the degree awarded to me on the basis of the present dissertation.

October 25, 2016
NIT Rourkela

Shashank Mouli Satapathy

Acknowledgment

I would like to express my earnest gratitude to supervisor of my doctoral program, Prof. Santanu Kumar Rath for believing in my ability and allowing me to work on the challenging domain of software effort estimation. His profound insights have enriched my research work. The flexibility of work, he has offered me has deeply encouraged me producing this research work. He is always being a source of support and motivation for bringing out quality in work. He has been supportive more than a professor and extended parental guidance during my research work.

I am very much indebted to the Doctoral Scrutiny Committee (DSC) members Prof. S. K. Patra, Prof. D. P. Mohapatra, Prof. P. M. Khilar and Prof. S. Das for their time to provide more insightful opinions into my research. Besides that, I am also thankful to all the Professors and faculty members of the department for their in-time support, advice and encouragement. I do acknowledge the academic resources that I have received from NIT Rourkela. I also thank the administrative and technical staff members of the Computer Science Department for their in-time support.

My hearty thanks goes to Mr. Ashish Kumar Dwivedi for his suggestions and thoughtful support in my decision making during the entire period of carrying out the research. He and his family are just like my family away from home. My sincere thanks to to all my fellow research colleagues Mukesh, Barada, Abinash, Lov, Aditi at National Institute of Technology Rourkela for their active or hidden cooperation.

I would conclude with my deepest gratitude to my parents, parents-in-law and all my loved ones. My full dedication to the work would have not been possible without their blessings, unconditional love, trust, and moral support. My special thanks to my beautiful and loving wife, Saswati and my son, Swastik. Their love, patience, support and understanding have lightened up my spirit to bring out quality. Understanding me best, Saswati has been my best friend and great companion, loved, supported, encouraged, entertained, and helped me to get through this agonizing period in the most positive way. This thesis is a dedication to them who did not forget to keep me in their hearts and all my loved ones, when I could not be beside them.

October 25, 2016
NIT Rourkela

Shashank Mouli Satapathy
Roll Number: 512CS104

Abstract

Estimation of effort for the proposed software is a standout amongst the most essential activities in project management. Proper estimation of effort is often desirable in order to avoid any sort of failures in a project and is the practice to adopted by developers at the very beginning stage of the software development life cycle. Estimating the effort and schedule with a higher accuracy is a challenge that attracts attention of researchers as well as practitioners. Predicting the effort required to develop a software to a certain level of accuracy is definitely a difficult assignment for a manager or system analyst, when the requirements are not very clearly identified. Effort estimation helps project managers to determine time and effort required for the successful completion of the project. In order to help the organization in developing qualitative products within a planned time frame, the job of appropriate software effort estimation is of primary requirement. For measuring the cost and effort of software development, traditional software estimation techniques like Constructive Cost Estimation (COCOMO) model and Function Point Analysis (FPA) have not been proved very much satisfactory, because of uncertainties associated with parameters such as Line Of Code (LOC) and Function Point (FP) respectively, used for procedural programming concept. The procedural oriented design splits the data and procedure, whereas accepted practice of present day i.e., the object-oriented design combines both of them.

Since class and use case are the basic logical units of an object-oriented system, the use of Class Point (CP) and Use Case Point (UCP) approach to estimate the project effort helps to get more accurate result. For projects based on the aspect of Web Engineering, effort estimation practice is identified as a critical issue. Considering these facts, there is a strong need for formal estimation of web-based projects, which can be accomplished by the help of International Software Benchmarking Standards Group (ISBSG) dataset. Similarly, in case of agile projects, Story Point Approach (SPA) is used to measure the effort required to implement a user story. By adding up the estimates of user stories which were finished during an iteration (story point iteration), the project velocity is obtained. The dataset related to CP, UCP and SPA are collected from previous projects mentioned in few research articles or from industries in order to assess the results.

In order to create results of estimation with more accuracy, when managing issues of complex connections in the middle of inputs as well as yields, and where, there is a distortion in the inputs by high noise levels, the application of machine learning (ML) techniques helps to bring out results with more accuracy. A number of past research

studies indicate that no single technique turns out to be the best for all cases. This is because of the dependency of system's execution altogether on the predicted function types, variations in properties of collected data, number of tests, noise ratio and so on. Hence the use of ML techniques in order to cope with issues arises in real-life situation is considered to be worthwhile. The research work carried out here presents the use of various ML techniques for software effort estimation using CP, UCP, Web-based and SPA approaches. The ML techniques are implemented taking into consideration of related dataset to predict the required effort.

The CPA is implemented using different ML techniques, i.e., Stochastic Gradient Boosting (SGB) and Support Vector Regression (SVR) kernels. Similarly, the UCP is implemented using ML techniques i.e., Random Forest (RF) and SVR Kernel. The techniques are implemented by taking into consideration of dataset based on one hundred forty nine number of projects on UCP collected from three different sources. Keeping in mind the end goal to enhance the efficiency of evaluating the effort required to develop web-based applications, certain ML techniques such as Decision Tree (DT), SGB, RF and SVR Kernel are employed on them. The SPA is implemented using ML techniques i.e., RF and SVR Kernel techniques. The dataset based on twenty one number of projects related to SPA are used for implementation.

In order to obtain convincing results in estimating software effort, the data obtained from previous projects help as a guidance and input to future estimation. Several methodologies have been proposed by researchers and practitioners for software effort estimation purpose. However, the CP, UCP and SPA are one of the various effort estimation models which are used in the proposed research work because of their characteristics such as simplicity, fastness and accurateness to a certain degree. Different ML techniques are employed on the CP, UCP, Web and SPA dataset collected from different sources in order to improve the accuracy of the prediction. Results obtained from applying different ML techniques are compared among themselves and with the results obtained from the existing results available in the literature, in order to assess their performances separately. On the basis of analysis of results obtained from each approach, it may be concluded that SVR RBF kernel based effort estimation technique yields better performance over other techniques used in this study for the considered dataset.

Keywords: ***Class Point Approach***; ***Use Case Point Approach***; ***Story Point Approach***; ***Web-based Applications***; ***Software Effort Estimation***; ***Machine Learning Techniques.***

Contents

List of Figures

List of Tables

List of Acronyms/ Abbreviations

SEE	Software Effort Estimation
CPA	Class Point Approach
UCP	Use Case Point Approach
ISBSG	International Software Benchmarking Standards Group
SPA	Story Point Approach
ML	Machine Learning
DT	Decision Tree
SGB	Stochastic Gradient Boosting
RF	Random Forest
RBF	Radial Basis Function
SVM	Support Vector Machine
SVR	Support Vector Regression
MAE	Mean Absolute Error
MSE	Mean Square Error
MMRE	Mean Magnitude of Relative Error
MMER	Mean Magnitude of Error Relative to the estimate
RMSE	Root Mean Square Error
NRMSE	Normalized Root Mean Square Error
PRED	Prediction Accuracy
SLIM	Software Life-cycle Management
FP	Function Point
COCOMO	Constructive Cost estimation Model
SLOC	Source Line of Code
IFPUG	International Function Point Users Group
NEM	Number of External Methods
NSR	Number of Services Requested
NOA	Number of Attributes
FPA	Function point Approach
UML	Unified Modeling Language

TUCP	Total Unadjusted Class Point
TCF	Technical Complexity Factor
ACP	Adjusted Class Point
UAW	Unadjusted Actor Weight
UUCW	Unadjusted Use Case Weight
EF	Environmental Factor
OOB	Out-of-Bag
AFP	Adjusted Function Point

Chapter 1

Introduction

Estimation of effort is considered to be a primary activity under the broad aspects of software project management, which is defined as the process of planning and controlling the development of a system at an optimal cost meeting the right set of requirements. It is an acknowledged fact that a good number of software fail due to faulty project management practices. Each year billions of dollars are wasted on entirely preventable mistakes. As per Robert N. Charette [1], the various common factors behind the failure of a software project are:

- Unrealistic or unarticulated project goals
- Inaccurate estimates of needed resources
- Badly defined system requirements
- Poor reporting of the project's status
- Unmanaged risks
- Poor communication among customers, developers, and users
- Use of immature technology
- Inability to handle the project's complexity
- Sloppy development practices
- Poor project management
- Stakeholder politics
- Commercial pressures

Therefore, it is quite necessary to adhere to key aspects of software project management activities. The software project estimation is considered as the most difficult and challenging task among all these features. Project estimation involves estimation of size, effort, cost, time, and staffing. For any software development project, the size of the product is often estimated at the very beginning stage. Taking input of the size of software, the effort needed are identified. From effort estimation, product duration and cost are found out.

Software size estimation is an important feature in order to determine the effort required to develop a software product. It is the methodology of anticipating the most practical measure of exertion (conveyed as individual hours or capital) needed to create or keep up development tasks in light of inadequate, questionable and uproarious data. Software Effort Estimation (SEE) is the procedure of foreseeing the most sensible utilization of effort required in order to develop or maintain software. SEE is the activity of estimating the total effort required to complete a software project [2]. Effectively assessing the effort required in order to develop a software product is of fundamental significance in order to sustain competitiveness in the market. Both under and over-estimation prompts undesirable results for the organizations. Under-estimation may bring about overwhelms in budget and schedule, which consequently may bring about the cancellation of projects; in this way, squandering the whole effort spent until that point. Over-estimation may bring about promising projects not to be subsidized; consequently, hurting the organizational capabilities. The process of effort estimation needs to be optimized because proper estimates are necessary both on the developer side as well as client side. On the developer side, estimates help in planning the development and monitoring the progress. While on the client side, they are used for negotiating contracts, setting completion dates, prototype release dates etc. However, as indicated in the research work reported by the Brazilian Ministry of Science and Technology-MCT, just 29% of the organizations fulfilled size estimation and 45.7% achieved software effort estimation. So the research work on effort estimation of proposed software has invited attention of a number of practitioners and theoreticians.

In the year 2013, the Standish Group Chaos Manifesto [3] states that 43% of IT projects were delivered late, over budget, and/or with less than the required features and functions. This indicates that the role of project management is being increasingly accepted as a more important aspect for sustainability [4,5]. The International Society of Parametric Analysis (ISPA) recognized the principle purposes behind failures of a majority of softwares [6, 7]. These reasons can be abridged as follows:

- Lack of understanding the requirements

- Improper software size estimation
- Lack of evaluation of the staffs expertise level

Another Standish report [8] outlines different principal factors, that expedite the failure of a software project such as:

- Realistic estimation
- Uncertainty in requirements of system and software
- Lack of skilled estimators
- Limitation in Budget
- Optimized software estimation process
- Lack of historical data
- Failed to consider historical data

In a nutshell, it is observed from the above parameters that numerous software projects fizzle due to incorrectness in software estimation process and poor understanding or inadequacy of the prerequisites. Hence, to obtain right kinds of results in estimating software effort, it is essential to consider the above issues and try to resolve them as much as possible. In the present day scenario, the object-oriented concept is the accepted practice of software development. As *class* and *use case* are the basic logical unit of an object-oriented system, the use of Class Point Approach (CPA) and Use Case Point Approach (UCP) to estimate the project effort help to guide the estimator in a more meaningful way. Web-based software projects are different than conventional object oriented projects, and hence the task of estimation for these projects is a complex one. As per Reifer [9], effort estimation models, which are helpful for conventional software development, are not extremely precise for effort estimation of web-based software development.

For effort estimation of web applications, the dataset of past web development projects are collected from ISBSG [10] dataset. Similarly, in case of agile projects, Story Point Approach (SPA) is used to measure the effort required to implement a user story. By adding up the estimates of user stories that were finished during an iteration (story point iteration), the project velocity is obtained. The efficiency of the models obtained using CPA, UCP, Web and SPA can be improved by employing certain intelligent techniques on them. The proposed research study considers the application of various machine learning (ML) techniques such as Decision Tree (DT), Stochastic

Gradient Boosting (SGB), Random Forest (RF) and Support Vector Regression (SVR) kernel methods over CPA, UCP, Web and SPA datasets in order to improve their prediction accuracies. These datasets are chosen by based on their contents and its relevance in order to employ effort estimation process on those dataset. The Class Point dataset are collected from [140], the UCP dataset are collected from 3 different sources, which includes dataset from industries and some are available for educational research purpose. The entire web dataset are collected from ISBSG repository and the SPA dataset are collected from [97]. The detailed description about these dataset are presented in the contributory chapters. The results of various models obtained after applying machine learning techniques are compared with each other as well as with the results available in the literature, in order to assess their performance.

1.1 Motivation

The motivation for this thesis is essentially to provide the estimating community with a fresh approach to the estimation problem, which might complement present practices. The main reasons for this are:

i) **Unimpressive results from algorithmic models:** Numerous empirical studies have been carried out by a number of authors in literature on the accuracy of algorithmic models. But somehow, the over-riding trend is inaccuracy and inconsistency. It may be possible to explore techniques other than algorithmic models in order to build effort prediction systems. One of the major problems with the use of algorithmic models is that they are dependent on quantifiable inputs. This often renders them ineffective during the early stages of a software project's conception. More appropriate approaches need to be found which can make estimates using the type of data those are present during the early stages of a project.

ii) **Lack of appropriate techniques for estimation of softwares developed using object-oriented methodology:** Object-oriented methodology is an approach of software development in the present-day scenario. But function point and COCOMO are the approaches which are still popular in the industries for effort estimation of object-oriented softwares. These techniques mostly depend on lines of code, which is obtained from the coding phase of software development life cycle (SDLC). Hence, for effort estimation during early stage of software development, i.e., starting with requirement analysis and design phase, more concentration should be given to estimate the effort of object-oriented softwares from UML diagrams.

iii) **Absence of applicable procedures for estimation of effort required to develop web-based applications:** Web-based software projects being considered in the present-day scenario are different from conventional object-oriented projects, and hence the task of estimation for this category is a complex one. Effort estimation models, which are helpful for conventional software development, are not extremely precise for effort estimation of web-based software development, because traditional effort estimation techniques are not adequate to capture specific features of the development which can influence the size and effort required in the development of web-based applications.

iv) **Unavailability of proper estimation techniques for softwares developed using agile methodologies:** Agile methodologies are gaining popularity year by year in software development industries. But due to lack of proper estimation techniques for softwares developed using agile methodology, failure rates are also more. Moreover, a number of agile methodologies such as scrum, extreme programming, lean programming etc. are followed by different industries for development of their softwares. Hence, it is quite difficult to propose a single estimation technique for softwares developed using different agile methodology.

1.2 Problem Statement

It has been observed from earlier research that, almost one-third number of projects surpass their budget and are conveyed late. Two-third number of projects invade their original estimates. It is an exceptionally troublesome assignment for a manager or system analyst to anticipate with much correctness the effort required to develop a software, when a number of external parameters such as unclear project definition, technological uncertainty, implementation complexity, team experience etc. [11] play a significant role. Hence, project managers usually are not able to determine truly, how much time and manpower a successful project needs. However, to help the organization in developing qualitative products inside planned period during the early stage of SDLC, legitimate estimation of software effort is essential.

1.3 Research Objective

This section indicates the progress stepped towards the above discussed state-of-the-art issues. The objectives of the research work outlined in this thesis are as follows:

1. To estimate the effort required to develop an object-oriented software utilizing

class point approach and improve the prediction accuracy of the result using different machine learning techniques.

2. To propose different machine learning techniques based effort estimation model for object-oriented softwares using use case point approach.

3. To assess the effectiveness of applying machine learning techniques for effort estimation of web-based applications and validate the result using industry dataset.

4. To analyze and compare the application of different machine learning techniques for effort estimation process of softwares developed using scrum based agile methodology.

Hence the overall research objective of this thesis is to estimate the effort of a software product using Class Point (CP), Use Case Point (UCP), Web and Story Point (SP) approaches. Then optimization of various parameters has been achieved using various ML techniques to obtain better prediction accuracy. Finally, the prediction accuracy obtained using different ML techniques have been compared in order to access their performance.

1.4 Machine Learning Techniques Used

The following machine learning techniques are applied over the various datasets considered to calculate the effort of a software product. The decision about choosing a machine learning technique for implementation purpose in the proposed research is performed based on the past research study done in the literature survey [12–15, 66]. Many researchers are applied some of the following machine learning techniques for their research purpose earlier. But none of these techniques are applied earlier for effort estimation using CP, UCP, Web and SP datasets. Every proposed contribution also describes a detailed presentation about the result obtained using these techniques for their corresponding dataset. Each contribution also depicts the in detailed comparison of these techniques with earlier result obtained from literature in order to access their performance.

1.4.1 Decision Tree Technique

A Decision Tree (DT) is an intelligent model characterized by a binary tree that illustrates the prediction of a dependent variable using a set of predictor variables. The primary DT model was proposed by Morgan and Sonquist in 1963 and was called

Automatic Interaction Detection (AID) [16]. This perspective was developed further by the THAID program in 1973 [17]. The fundamental point of interest of a DT model is that it can help a novice to investigate the master plan of a specific issue. In any case, the fundamental inconvenience of a DT model is that every node is optimized locally rather than global optimization of the entire tree. Besides, DT models may experience the ill effects of the over-fitting issue, and in addition from giving good accuracy in contrast with different models.

1.4.2 Stochastic Gradient Boosting Technique

The Stochastic Gradient Boosting (SGB) technique is also called as the Tree-boost model [18]. "Boosting" technique considers a function iteratively in a series and combines the output of each function with a weighting coefficient in order to minimize the total error of prediction and increase the accuracy. The mathematical representation of the SGB algorithm can be written as

$$F(y) = F_0 + C_1 \times T_1(y) + C_2 \times T_2(y) + ... + C_M \times T_M(y) \tag{1.1}$$

where $F(y)$ is the estimated target value and F_0 is the initial value for the series. Vector y is used to represent the pseudo-residual values remaining at this point in the series. To fit the pseudo-residuals, a series of trees $T_1(y)$, $T_2(y)$ etc. are used. C_1, C_2 etc. are coefficients of the tree node estimated values that are calculated using the SGB technique.

Often it is observed that, an individual tree consists of eight terminal nodes with depth level 3. Hence, it is fairly small. But, the full SGB model is built with large numbers of these small trees. Beginning with the first tree, successive trees are fitted to the data. The residuals (error values) from the preceding tree are fed into the next tree in order to reduce the error. After repeating the process for a chain of trees, the final predicted value is obtained by the summation of the weighted contributions of individual trees. The Tree-boost method uses the *Huber-M loss function* for regression. Residuals falling under the *Huber's Quantile-Cutoff* are squared before use. In other cases the absolute values are used.

Literally *"Stochastic"* means a random percentage of training data points i.e., 50% is recommended, are used for each iteration instead of all. In order to delay the learning process and elongate the length of the series, a *shrinkage factor* (between 0 and 1) is multiplied to each tree in the series. In return the increased length compensates for the shrinkage. This activity improves the prediction values. An *Influence Trimming Factor* is applied to optimize the process, as it allows the rows with small residuals to be excluded.

1.4.3 Random Forest Technique

Random Forest (RF) is an ensemble learning technique used for classification and regression purposes [19]. It builds a number of decision trees during training period and chooses the final class by selecting the mode of the classes generated by distinctive trees. To obtain better results which are competitive than the results from individual decision tree models, ensemble model combines the results from different models of similar type or different types.

The concept behind the RF is that it generates a number of classification trees with the help of a random vector 'λ' and an input vector 'x'. A random vector 'λ_k' is produced for the kth tree, which is autonomous of the previous random vectors $\lambda_1, ..., \lambda_{k-1}$ with equal distribution. A tree is developed using the training set and λ_k, which generates a classifier h(x, λ_k), where 'x' is an input vector. To categorize new object from an input vector, the input vector 'x' is jotted down along with each of the trees in the forest. Each tree provides a classification by voting for that class. Then, the classification having the maximum number of votes among all the trees in the forest is chosen. In case of regression, the prediction accuracy of the forest is obtained by taking the average of predictions for individual tree.

RF for regression purpose are created by developing trees relying upon a random vector λ, which is specified as the tree predictor h(x, λ) that undertakes numerical data instead of class labels. The output produced by the predictor is $h(x)$ and the actual effort value is Y. For any numerical predictor $h(x)$, the generalized mean-squared error is calculated as

$$E_{x,Y}(Y - h(x))^2 \tag{1.2}$$

By calculating the average value obtained over k trees h(x, λ_k); the RF predictor is modeled.

1.4.4 Support Vector Regression Technique

Support Vector Machines (SVM) are a category of learning machines, helpful for implementing the structural risk minimization inductive principle in order to obtain a good generalization on a limited number of learning patterns. A version of SVM for regression was proposed by Vapnik et al. [20] in 1996. This method is called as *support vector regression (SVR)*. It is very often observed that any neural networks suffers from two major drawbacks. First of all, neural networks often converge on local minima rather than global minima. Secondly, neural networks often over-fit which means, if training on a pattern goes on too long, then it may consider noise as part of pattern. SVR technique does not suffer from either of these two drawbacks

and have the advantages due to which it can be successfully used for regression task. Firstly it has a regularization parameter, which makes the user consider staying away from over-fitting. Furthermore it utilizes the kernel trick, so that expert knowledge regarding the issues can be build through optimizing the kernel. Thirdly a SVR is characterized by a convex optimization issue. Ultimately, it is an estimate to a bound on the test error rate, and there is a significant assemblage of hypothesis behind it, which proposes it ought to be a smart thought.

Suppose, for a given training data $(x_1, y_1), \ldots, (x_l, y_l)$, where $x \in \mathbb{R}^n$ denotes the space of the input patterns and $y \in \mathbb{R}$ denotes its corresponding target value, the goal of regression may be identified as to find the function $f(x)$ that best models the training data. For the case of nonlinear regression, $f(x) = \langle w, \phi(x) \rangle + b$, where ϕ is a nonlinear function which maps the input space to a higher (maybe infinite) dimensional feature space and $\langle ., . \rangle$ denotes the dot product in $\mathbb{R}^n$. In SVR, the weight vector 'w' and the threshold 'b' are chosen to optimize the following problem [21].

$$\min_{w,b,\xi,\xi^*} = (1/2 w^T w + C \sum_{i=1}^{l} (\xi_i + \xi_i^*))$$

$$subject\ to \quad \begin{cases} (\langle w, \phi(x_i) \rangle + b) - y_i \leq \epsilon + \xi_i, \\ y_i - (\langle w, \phi(x_i) \rangle + b) + \leq \epsilon + \xi_i^*, \\ \xi_i, \xi_i^* \geq 0. \end{cases} \tag{1.3}$$

where $C > 0$ is the *penalty parameter* of the error term. ξ and ξ^* are called *slack variables* and measure the cost of the errors on the training points. ξ measures deviations surpassing the target value by more than ϵ and ξ^* measures deviations which are more than ϵ, however underneath the target value [12]. Intuitively, a kernel is just a transformation of the input data that allows the user to process it more easily. It helps in performing certain calculation faster which otherwise would involve computations in higher dimensional space. It allow us to do stuff in infinite dimensions! Sometimes going to higher dimension is not just computationally expensive, but also impossible. Then kernel provides a wonderful way to deal with this issue.

$K(x_i, x_j) = \phi(x_i^T \phi(x_j))$ is called the *kernel function*. Basically four varieties of kernels are available, which can be identified as:

- **Linear Kernel:**
  $K(x_i, x_j) = x_i^T x_j.$
- **Polynomial Kernel:**
  $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0.$

- **Radial Basis Function (RBF) Kernel:**
  $K(x_i, x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2), \gamma > 0.$
- **Sigmoid Kernel:**
  $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r).$

Here γ, r, and d are *kernel parameters*. Selecting a specific kernel type and kernel function parameters is typically in view of application-domain knowledge; furthermore ought to reflect conveyance of input values of the training data. In epsilon-SV regression [21], the goal is to find a function $f(x)$ that has at most ϵ deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible. In other words, errors less than ϵ are ignored and considered as zero. But errors larger than ϵ are measured by variable ξ and ξ^*. The following tunable parameters [22] have been used while implementing support vector regression.

- **param**: This is a string which specifies the model parameters. For regression model, a typical parameter string may look like, *'-s 3 -t 2 -c 20 -g 64 -p 1'* where
  - **-s**: svm type,
  - **-t**: kernel type
  - **-c**: penalty parameter C of epsilon-SV regression.
  - **-g**: width parameter γ
  - **-p**: ϵ for epsilon-SV regression.

The value of parameter 's' ranges from 0 to 5 and the default value is 0. For epsilon-SV regression, the parameter 's' is assigned with value 3. The 't' value ranges from 0 to 3 for different types of kernel. In this case, the value can be 0, 1, 2 or 3 for linear, polynomial, RBF and sigmoid kernel respectively. The default value for 't' is 2. Similarly, the value of parameter 'c' will be calculated as the difference between maximum and minimum value of actual effort used to train the model. The default value is 1. The parameter 'g' value signifies width parameter i.e., it set γ in various kernel function. The default value is 1. Lastly, the value of parameter 'p' set the ϵ in loss function of epsilon-SVR. The default value for parameter 'p' is 0.1.

1.5 Evaluation Criteria

The evaluation of the performance obtained using various machine learning techniques is carried out by employing different criteria. These criteria are used to measure the performance of ML techniques in terms of their generated error value and prediction

accuracy. One criteria is also used in order to test the statistical significance among various ML techniques. Few other criteria are also used to evaluate the effect size i.e., trying to evaluate the magnitude of treatment effect. The statistical significance test are dependent on sample size; where as effect size test is independent of sample size. The detailed description of the above mentioned criteria are outlined below [23–26]:

- The **Mean Absolute Error (MAE)** is the average of the absolute errors between the actual and the predicted effort as shown in Equation 1.4.

$$MAE = \frac{1}{TP}\sum_{i=1}^{TP}|AE_i - PE_i| \tag{1.4}$$

  where
  AE_i = Original effort value collected from the dataset for the i^{th} test data.
  PE_i = Output (predicted effort) obtained using the developed model for the i^{th} test data.
  TP = Total no. of projects in the test set.

- The **Mean Magnitude of Relative Error (MMRE)** can be obtained through the summation of Magnitude of Relative Error (MRE) over N observations

$$MMRE = \frac{1}{TP}\sum_{i=1}^{TP}\frac{|AE_i - PE_i|}{AE_i} \tag{1.5}$$

- The **Mean of Magnitude of Error Relative to the estimate (MMER)** is one of the criteria used for effort estimation models evaluation. MMRE and PRED(25) measure diverse properties of the distribution of 'z', where $z = predicted/actual$. In this manner, 'z' is thought to be a measure of precision, and insights, for example, MMRE and PRED(25) to be measures of properties of the distribution of 'z'. In this way, it is not surprising that the two measurements may seem to give conflicting results, on the off chance that they are utilized to assess alternative prediction systems. Hence, it is contended that MMER can provide higher accuracy than the Mean Magnitude of Relative Error (MMRE) [27, 28]. MMER is the mean of MER as shown in Equation 1.6.

$$MMER = \frac{1}{TP}\sum_{i=1}^{TP}\frac{|AE_i - PE_i|}{PE_i} \tag{1.6}$$

- The **Root Mean Square Error (RMSE)** is calculated as the square root of mean square error (MSE). MSE is calculated by finding out the mean of the

square of the difference between the actual and predicted effort value.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{TP} (AE_i - PE_i)^2}{TP}} \tag{1.7}$$

- The **Prediction Accuracy (PRED (x))** is PRED can be described as the average of the MAE's off by no more than x as shown in Equation 1.8.

$$PRED(x) = \frac{1}{TP} \sum_{i=1}^{TP} \begin{cases} 1 & if MAE_i \leq x \\ 0 & Otherwise \end{cases} \tag{1.8}$$

The accuracy of the estimates is directly corresponding to PRED(x) and conversely relative to MMER.

- The **Mann-Whitney U test** is a non-parametric test [29, 30], which is an alternative test to the independent sample t-test. It is used to compare two population means that come from the same population, having equal means or not. It allows two groups or conditions to be compared without making the assumption that values are normally distributed. It is used for equal sample sizes, and is used to test the median of two populations [31, 32]. Usually the Mann-Whitney U test is used when the data is ordinal. The procedure to calculate Mann-Whitney p-value is outlined below:

$$U = N_1 N_2 + \frac{N_2(N_2+1)}{2} - \sum_{i=N_1+1}^{N_2} R_i \tag{1.9}$$

where,
N_1 = First Sample Size
N_2 = Second Sample Size
R_i = Rank of the Sample Size
The maximum possible values of (R_i can be $N_1N_2 + \frac{N_2(N_2+1)}{2}$). While performing statistical significance test between two techniques using Mann-Whitney p-value, first Null and Alternate hypothesis is formed. For this study, the Null and Alternate Hypothesis is presented below:
Null Hypothesis (H_0): The two techniques are not different.
Alternate Hypothesis (H_1): The two techniques are different.
If the p-value is less than 0.05, the techniques are statistically significant at 95% confidence interval. Hence, the Null hypothesis should be rejected.

- In statistics, the **effect size** is a measure of the quality of the relationship between two variables in a statistical population, or a sample-based evaluation of

that quantity. There are three better-known ways available in order to calculate the effect size, such as Cohen's d, Glass's Δ and Hedges's g [33, 34]. Hedges's g approach is used when there is a difference in sample size. In this study, the sample sizes are not different. Hence, Cohen'd d and Glass's Δ approaches are taken into consideration in this study in order to evaluate the effect size. The Cohen's d is determined by calculating the mean difference between two groups and then dividing the result by the pooled standard deviation. The computation procedures for the approach is outlined below:

$$Cohen's\ d = \frac{M_1 - M_2}{SD_{pooled}} \tag{1.10}$$

where, M_1 and M_2 represents the mean of first and second sample. The value of SD_{pooled} can be calculated as:

$$SD_{pooled} = \sqrt{\frac{{SD_1}^2 - {SD_2}^2)}{2}} \tag{1.11}$$

In case the standard deviations of the two samples vary, then the homogeneity of variable assumption is abused; hence pooling the standard deviations is not proper. One arrangement is to embed the standard deviation of the control group into the condition in order to figured out Glass's Δ. The procedure to calculate Glass's Δ is outlined below:

$$Glass's\ \Delta = \frac{M_1 - M_2}{SD_{control}} \tag{1.12}$$

The rationale is that the standard deviation of the control group is untainted by the impacts of the treatment and consequently it will more meticulously emulate the population standard deviation. The quality of this presumption is specifically relative to the extent of the control group. The bigger the control group, the more it is liable to look like the population from which it was drawn. In this study, the first ML technique is assumed to be the experimental group and the second technique is assumed to be the control group.

As per the categorization made by Cohen [35], the effect size is broadly categorized into three categories i.e., small ($\simeq 0.2$), medium ($\simeq 0.5$) and large ($\simeq 0.8$). As mentioned by Cohen, a small effect size is one in which there is a genuine impact i.e., something is truly happening on the planet, however which must be seen through cautious study. A large effect size is an impact which is sufficiently enormous, and/or sufficiently steady, that might have the capacity to be seen with the naked eye. A large effect size is one which is extremely

significant.

1.6 Dissertation Layout

This thesis is organized into seven different chapters including Introduction chapter. Each chapter is discussed below in a nutshell:

Chapter 2: Literature Review

This chapter focuses on the state-of-art of various models for software effort estimation. The review has been performed in six sections with respect to objectives of the thesis. The first section of the survey highlights on the basic software effort estimation techniques. The second and third section highlights on various key aspects for effort estimation of object oriented software using class point and use case point approach accordingly. The fourth section of the chapter highlights on the survey of articles proposing various techniques for effort estimation of web applications. Agile software effort estimation process is an emerging area of research. The fifth section of the chapter highlighting the research work carried out earlier on the area of agile software effort estimation. The last section deals with presenting various articles, where different machine learning techniques are used for software effort estimation process along with their corresponding implications over the estimation accuracy result.

Chapter 3: Class Point Approach for Software Effort Estimation using Machine Learning Techniques

This chapter focuses on designing effort estimation models for object-oriented softwares based on class point approach using various machine learning techniques. Later, the chapter draws a comparative analysis for the results obtained from the different machine learning techniques based effort estimation models in order to assess their performance.

Chapter 4: Use Case Point Approach for Software Effort Estimation using Machine Learning Techniques

This chapter focuses on inspecting the application of machine learning techniques for software effort estimation based on use case point approach. Various machine learning techniques based effort estimation model have been proposed by considering the use case point dataset as input and compared in order to access their performance.

Chapter 5: Effectiveness of Machine Learning Techniques for Effort Estimation of Web Applications

In this chapter, machine learning techniques are applied for effort estimation for web-based applications. The dataset of applications based on web are collected from the International Software Benchmark Standards Group (ISBSG) repository in order to validate the result. Finally, the results are compared to draw the conclusion of the analysis.

Chapter 6: Story Point Approach for Agile Software Effort Estimation using Machine Learning Techniques

Agile software effort estimation is one of most important area of research nowadays. There are a number of techniques available for development of software using agile methodology such as Scrum, Extreme Programming, Lean etc. This chapter deals with highlighting the procedures developed for effort estimation of software developed using agile methodology, especially scrum based development. Finally, the obtained results are compared for further assessment.

Chapter 7: Conclusion

This chapter presents the conclusions drawn from the proposed work with accentuation on the work done. The limitations associated are highlighted. The extension for further research work in this direction has been explained at the end.

Chapter 2

Literature Review

Software Effort Estimation (SEE) is one of the important activities carried out before going ahead with development activities of proposed software. To deal with challenges in estimation of proposed software, various researchers and practitioners have proposed different approaches. This chapter presents a survey of various approaches for software effort estimation. The chapter has been divided into various sections. The section 2.1 presents the survey of various techniques proposed for basic software effort estimation. These include popular techniques such as algorithmic models i.e., SLIM, Function Point, COCOMO etc., expert judgment and estimation by analogy. Section 2.2 deals with presenting various articles related to class point approach based software effort estimation procedure. Section 2.3 presents the survey of articles dealing with use case point approach based software effort estimation. Section 2.4 surveys articles deal with effort estimation of web application. Similarly, section 2.5 presents articles providing procedures for agile software effort estimation. Finally, section 2.6 presents the survey of various articles focusing on various machine learning techniques for software effort estimation procedure.

2.1 Survey on Basic Software Effort Estimation Techniques

The Software Life-cycle Management (SLIM) model, which is otherwise called Putnam model was proposed by Lawrence Putnam in 1978 [36]. The SLIM depicts the effort and time required to complete the development of software of a specific size. The time-effort curve of Putnam model follows the Rayleigh distribution [37]. Function Points measure the functionality of a software as opposed to SLOC, which measures the physical component of a software. It was developed by Allan Albrecht in 1979 [38]. The International Function Point Users Group (IFPUG) [39] defines the stabdard procedure to be followed to count function points. The COnstructive COst MOdel

(COCOMO) is an algorithmic model utilized to anticipate software cost. It was produced by Barry Boehm in 1981 [40], and was known as COCOMO'81. COCOMO depends on regression model.

R. T. Hughes [41] has proposed a model based on expert judgment by a group of experts to utilize their experiences for estimation of a proposed software. The Delphi technique [42] can be used to provide communication and cooperation among experts. One of the major drawbacks of the expert judgment model is the lack of analytical argumentation, because of the frequent use of phrases, which is identified in [43]. Function Point approach and COCOMO experience the ill effects of the impediment of the need to align the model to every individual estimation environment combined with variable precision levels even after adjustment. Another approach is to utilize analogy based estimation strategy proposed by Shepperd et al. [44]. They have evaluated analogy approach with six distinct datasets drawn from a range of different environments and their approach is being claimed to outperform other methods. The main disadvantage of analogy method is that it requires considerable amount of computation. Walkerden and Jeffery [45] have compared few techniques for analogy-based software effort estimation with each other furthermore with a linear regression model. The outcomes demonstrated that human brains work superior than tools at selecting analogies for the considered dataset. Estimates based on their selections, with a linear size adjustment in accordance with the analogue's effort esteem, demonstrated more precise results than estimates based on analogues selected by tools, furthermore more exact than evaluations based on the simple regression model. Idri et al. [46] have proposed new and modified Analogy-based Software development Effort Estimation (ASEE) techniques and the detailed analysis of result showed that ASEE methods outperform the eight techniques with which they were compared, and tend to yield acceptable results especially when combining ASEE techniques combines with Fuzzy Logic (FL) or Genetic Algorithms (GA). Idri et al. [47] have also proposed a novel analogy-based technique, called 2FA-kprototypes, to foresee effort when software projects are depicted by a blend of numerical and categorical attributes and coordinated fuzzy k-prototypes calculation into the procedure of estimation by analogy. The estimation precision of 2FA-kprototypes was assessed and contrasted with two techniques i.e., classical analogy-based technique and 2FA-kmodes utilizing four datasets. The outcomes acquired demonstrated that both 2FA-kprototypes and 2FA-kmodes perform superior than classical analogy-based technique.

Molokken and Jorgensen [48] abridged estimation knowledge by conducting a survey on software effort estimation. They found that most projects (60-80%) experience effort and/or schedule overruns. The estimation techniques in most

regular utilization of expert judgment is that there is no confirmation that formal estimation models lead to more correct assessments. The review likewise proposed that there is a lack of surveys including extensive analyses of the reasons for effort and schedule overruns. Magne Jorgensen [49] displayed seven rules for producing realistic software development effort estimates, which are derived from industrial experience and observational studies. By dissecting the rules, they found that assessing effort on the premise of expert judgment is the most widely recognized approach today, and the choice to utilize such procedures rather than formal estimation models can prompt more practical appraisals of software development efforts. Kocaguneli et al. [50] investigated the use of transfer learners for software effort estimation, when project needs adequate local data in order to make accurate prediction. They have utilized dataset based on 154 number of projects collected from two different sources to examine transfer learning between various time intervals and another dataset based on 195 number of projects collected from 51 different sources to give a proof on the estimation of transfer learning for customary cross-company learning issues. From the examination of the outcome, it is found that transfer learning is a promising research direction that exchanges applicable cross data between time intervals and areas.

Whigham et al. [51] have proposed an Automatically Transformed Linear Model (ATLM) as a reasonable baseline model for examination against software effort estimation strategies. ATLM is a basic model, yet performs well over a range of various project types. Additionally, ATLM might be utilized with mixed numerical and categorical data and requires no parameter tuning. It is also deterministic in nature which means that results obtained are amenable to replication. They have suggested that ATLM should be used as a baseline of effort prediction quality for all future model comparisons in SEE. Gonzalez et al. [52] have performed a systematic mapping study over 107 number of papers that use International Software Benchmarking Standards Group (ISBSG) data for effort estimation. They described the usage of ISBSG variables for filtering, as dependent variables, and as independent variables and identified 20 variables (out of 71) mostly used as independent variables for effort estimation. By analyzing their study, they proposed guidelines for researchers to make informed decisions about which different ISBSG variables to be selected for their effort estimation models.

2.2 Survey on Class Point Approach

During the calculation procedure of adjusted class point as identified by Costagliola et al. [53], two measures, Class Point 1 (CP1) and Class Point 2 (CP2), are utilized. CP1 is figured utilizing two measures, Number of External Methods (NEM) and Number

of Services Requested (NSR); whereas CP2 is ascertained by utilizing an alternate metric as a part of expansion to NEM and NSR, Number of Attributes (NOA). They have observed that the prediction accuracy of CP1 and CP2 under the class point approach were 75% and 83% respectively. They drew this conclusion by conducting an experiment on a dataset with forty projects. Zhou and Liu [54] have extended this methodology by including an alternate measure CP3 and considered twenty four attributes rather than the eighteen acknowledged by Gennaro Costagliola et al. By utilizing this methodology, they watched that the performance of CP1 and CP2 stay unaltered, although the number of characteristics changed. Kanmani et al. [55] have utilized the same CPA with the ANN model for mapping CP1 and CP2 into the assessed software development effort and observed that the prediction accuracy for CP1 was enhanced to 83% and CP2 to 87%. Kim et al. [56] have presented some new meanings of class point to interpret system's architectural complexity in an improved way. They have utilized various additional parameters along with NEM, NSR and NOA to compute the total number of adjusted class point value.

Kanmani et al. [57] have introduced a novel technique to utilize the CPA with fuzzy logic by embracing the subtractive clustering technique for computing effort and contrasted it with the result acquired from the ANN. They observed that the fuzzy system focused around the subtractive clustering technique outperforms ANN. Kapoor and Pandey [58] have applied fuzzy logic technique along with class point approach for size estimation of object oriented products. Fuzzy logic allows a gradation of values instead of discrete sets, which in turn allows it to be more tolerant to uncertainty, imprecision, partial truth and approximation and thus achieve tractability, robustness and low cost solution. They have proved that fuzzy class point approach yields better results than traditional methods. .

2.3 Survey on Use Case Point Approach

Issha et al. [59] have investigated the evolution of three different use case model-based software effort estimation techniques. The correctness of the proposed techniques is verified using a wide range of software projects. Nassif et al. [24] have presented two models i.e., log-linear regression (LLR) model and Multi-Layer Perceptron (MLP) model, based on use case point to compute the software development effort focused around use case diagrams. By analyzing the outcome, they have proved that the MLP model performed better than other models for smaller projects; however, the LLR model outperforms other models for large size projects. Nassif et al. [60] have presented a novel regression technique for effort estimation of a given software focused around the UCP. They proposed a software effort estimation equation that considers

the non-linear relationship between software effort and size, and in addition on the impacts of projects complexity and productivity. Results show that the accuracy of estimating the software development effort gets enhanced by 16.5% than the result obtained using Karner's model. Urbanek et al. [61] have analyzed the statistical value of Use Case Points method parameters to find any parameters in Use Case Points method, which can be omitted from the calculation and the results may turn out to be better, while analytical programming for effort estimation is being considered. From the result, it was observed that the accuracy of Use Case Points method is improved if and only if UUCW parameter is present in the calculation.

Nassif et al. [62] have extended this process by applying Mamdani fuzzy inference system with regression model to enhance the estimation accuracy and found 10% improvement over Karners model and 6% over Schneiders model. Nassif et al. [63] also applied Sugeno fuzzy inference system with regression model to enhance the estimation accuracy and found 11% improvement in the Mean Magnitude of Relative Error (MMRE) result over Karners model and 7% over Schneiders model. Nassif et al. [64] have proposed an Artificial Neural Network (ANN) model to anticipate software effort from use case diagrams based on the UCP model with the assistance of a dataset based on 240 number of projects and obtained an improved result than other regression models. A. B. Nassif [65] has also proposed some other techniques using fuzzy logic and ANN to enhance the correctness of the UCP model and achieved up to 22% improvement in prediction accuracy result over Karners model.

Nassif et al. [66] have used a tree boost (Stochastic Gradient Boosting) model to estimate the effort required to develop a software product focused around UCP method using a eighty four project dataset and achieved improved results. Saroha and Sahu [67] have reviewed various techniques used for software effort estimation and also provides comprehensive analysis of various tools and frameworks developed for efforts estimation based on Use Case Point (UCP) model. They have observed that the analyzed tools provide opportunity to consider some other factors, which may affect project delivery and help in providing a better estimate of project effort than the existing ones. Silhavy et al. [68] have presented a new size estimation method known as Algorithmic Optimisation method for estimate size of software engineering projects. This method is based upon use case point and multiple least square regression and is derived into three phases. They have observed that the proposed method performs approximately 43% better than use case point approach based on their magnitude of relative error score.

Periyasamy and Ghode [69] have extended the original UCP model with additional information obtained from use case narratives. They classified actors into seven distinct groups. Additionally, the authors proposed new weights for use cases. The

weight of an use case is resolved in light of the quantity of relationship amongst actors and the use cases. Anda et al. [70] have provided guidance for other organizations who want to improve their estimation process applying use cases considering three industrial case studies. Results demonstrate that the direction gave by the use case point strategy can bolster expert knowledge in the estimation procedure and the configuration of the use case models strongly affects the assessments. Sergey Diev [71] has presented a number of real world situations taking into account the experience accumulated during deployment of the UCP in a product development department of a noteworthy financial establishment. The author exhibited that in order to get sensibly precise estimates, it is wanted to reflect in used case models a few aspects of the existing application and of the present project. Likewise, the author recommended a few elucidations of the idea of use case transaction and frameworks some approaches to bolster use case models consistency inside and crosswise over projects.

Ajitha et al. [72] have built up a neural system model to evaluate the size of software utilizing Use Case Point approach. The outcomes are approved and a contextual analysis of Multi-Agent System and showed improvement over the existing ones. Iraji and Motameni [73] have presented an adaptive fuzzy neural network model to estimate the effort of object oriented software using Use Case size Point approach. Results indicate that the proposed approach possesses less error and worked more accurately than methods evolved earlier. Nassif et al. [74] have proposed an approach to calibrate the complexity weights of the use cases in the Use Case Points (UCP) model. They applied a neural network with fuzzy logic to tune the complexity weights. Saroha and Sahu [75] have proposed an enhanced Use Case Point model (Algorithmic model) to overcome the problems arise from using the existing Use Case Point (UCP) model by considering five software projects case studies and different evaluation criteria like MMRE, MMER, MSE, RMSE and PRED(x). It was evident that the result obtained from the proposed model outperforms the results obtained from existing UCP model.

2.4 Survey on Effort Estimation of Web Applications

Mendes et al. [76] have worked extensively on the aspects of size-measurements and cost drivers for early stage web effort estimation by taking the help of dataset based on 133 number of web projects. Results showed that the two most basic size measurements utilized for web effort estimation were "total number of web pages" (70%) and "which functionality to be given by the application"(66%). Emilia Mendes [77] has employed four different approaches for estimation of effort required to develop

web-based applications: Forward Stepwise Regression (FSR), Bayesian Networks (BNs), Case Based Reasoning (CBR) and Classification and Regression Trees (CART) to get the estimated effort and compared them. Mendes and Mosley [78] have also compared several BN models using a cross-company dataset for effort estimation in web developments. The developed models' various performance parameters were also compared to mean and median-based effort models, MSR and CBR. Corazza et al. [13] have investigated the use of Tabu Search meta-heuristic methodology in blend with SVR to choose a suitable subset of parameters to be utilized for web effort estimation using the same database and obtained promising results.

Ferrucci et al. [79] have inquired the viability of Tabu Search in assessing the effort required to develop web-based applications with the help of Tukutuku cross-company database and obtained encouraging results. Elyassami and Idri [80] have investigated the effectiveness of applying Fuzzy ID3 decision tree technique for software effort estimation purpose. This technique is outlined by incorporating the standard principles of fuzzy set-theoretic concepts into the ID3 decision tree. Corazza et al. [14] have investigated the viability of SVR for web effort estimation utilizing a cross-company dataset and thought about diverse SVR designs taking a gander at the particular case that exhibits the best execution. The dataset utilized for validation was the Tukutuku database and results demonstrated that the SVR RBF outperforms others. Martino et al. [81] have enquired the potency of the Web Objects measure as an indicator of We-based software development effort. The effectiveness of the Web Objects measure as indicator of Web application development effort was confirmed, when assembled with Ordinary Least-Squares Regression (OLSR) and WebCOBRA, and this is true even when using CBR. It was observed that the Web Objects method yields better results than the FPA method when assembled with OLSR and Web-COBRA.

Ferrucci et al. [82] have studied the suitability of web effort estimation models developed with the help of cross-company dataset and compared it with the model based on single-company dataset. They have performed the validation of the models considering dataset based on 195 number of web projects obtained from the Tukutuku database as input. Results proved that although the prediction accuracy value of the model obtained using cross-company dataset was not more impressive than that of single-company models; but by applying the filtering mechanism, the prediction accuracy value can be improved significantly. Corona et al. [83] have presented a new methodology for developing a web effort estimation model with a content management framework (CMF) and performed the experimental validation using dataset of nine numbers of projects as provided by three different Italian software companies.

Kocaguneli et al. [84] have presented the effectiveness of applying ensembles of

effort estimation techniques. They generated ninety number of solo methods, which were applied to twenty number of datasets and the results were evaluated using seven numbers of error measures. It is observed that authors have combined few solo methods to generated twelve number of multi methods. From the analysis of the result, they observed that no single effort estimation method can be identified to be the best, but there exists a suitable combination of such effort estimation methods, which may yield better results. Azhar et al. [85] have presented the use of ensembles of effort estimation techniques for web project data using two approaches i.e., replication of methodology and using Scott-Knott algorithm. The replication identified 16 number of techniques out of 90 number of solo estimation techniques on web project data from the Tukutuku dataset to build 15 ensembles; whereas the Scott-Knott algorithm identified 19 superior solo techniques that were used to build two ensembles. Results showed that ensembles of techniques outperformed solo estimation technique. The study carried out by Azhar et al. [85] is a production study of Kocaguneli's [84] work.

Matos et al. [86] have performed information examination utilizing Grounded Theory-based techniques to distinguish and join components influencing the effort estimation process of web applications and recognized four categories of factors. The factors available in each of these groupings affects the process of estimating effort for web applications. Matos et al. [87] have also extended the work to build the comprehension of web effort estimation by utilizing the same set of factors already identified in the previous article alongside the knowledge from experts to handle effort estimation process. They have recognized a sum of 90 number of variables which make an impact on effort estimation in web applications, out of which only 30 number of components were distinguished during extensive research study carried out with experts.

Nassif et al. [88] have provided a comparative analysis of results obtained by applying four different neural network models such as Multi-layer Perceptron (MLP), General Regression Neural Network (GRNN), Radial Basis Function Neural Network (RBFNN) and Cascade Correlation Neural Network (CCNN) for software effort estimation purpose. They observed that CCNN model outperformed other models based on 60 % of the dataset, where as RBFNN outperforms other models based on 40% of the datasets. They also proved that CCNN model is not statistically different from other models despite of its higher performance. Denis and Boris [89] have analyzed the possibility of using a combination of functional size and conceptual models for the purpose of web application development effort estimation. They have employed 19 web applications with their conceptual models to build an effort model using simple linear regression analysis and obtained promising results for web projects used in the model construction and validation process. Barabino et al. [90] presented a

new methodology, called *Web Framework Points*, in order to assess the effort required to develop web-based applications with Content Management Framework (CMF). They performed the validation of the results obtained from the research work by using dataset based on 29 number of projects, out of which 83% showed less than 25% of estimation error value.

Bhardwaj and Rana [91] portrayed the connections among size of a software, no. of software defects, productivity and efforts for web applications established utilizing the multiple linear regression technique on the data collected from ISBSG. Results suggest that in web-based projects the number of defects identified is directly proportional to the productivity. Therefore, less testing and rework effort will be required if project is planned with lower productivity. Minku et al. [92] examined the utilization of Dycom approach to evaluate to what degree Web effort estimation acquired utilizing cross-comapny (CC) datasets are viable in connection to the predictions got utilizing within-company (WC) data when unequivocally mapping the CC models to the WC context. A 125 number of web-based projects data 25 from eight distinctive organizations part of the Tukutuku database were utilized to build prediction models. They have benchmarked these models against baseline models (mean and median effort) and a WC base learner that does not advantage of the mapping. By dissecting the outcomes, it was evident that Dycom mostly accomplish comparable or preferred execution over a WC model while utilizing just 50% of the WC training data. Martino et al. [93] have empirically investigated the effectiveness of COSMIC over FPA for Web effort estimation. Two experimental studies have been performed by with the help of an industrial data set. Aftereffect of the principal study uncovered that, COSMIC was essentially more exact than FPs in assessing the development effort for considered dataset. The second study uncovered that the viability of the investigated two-stage process fundamentally relies on the utilized conversion equation.

2.5 Survey on Story Point Approach for Agile Software Effort Estimation

Keaveney and Conboy [94] have investigated the applicability of conventional estimation techniques towards agile development approaches by underscoring on the case studies of agile method utilized within diverse organizations. Coelho and Basu [95] have described the steps followed in story point-based method for effort estimation of agile software and highlighted the areas which need to be looked into for further research. Andreas Schmietendorf et al. [96] have provided an investigation about estimation possibilities, especially for the extreme programming paradigm.

Ziauddin et al. [97] have developed an effort estimation model for agile software

projects, where model was fine-tuned with the help of the empirical data acquired from twenty one software projects. Usman et al. [98] have provided a detailed overview of the state of the art in the area of effort estimation in agile software development. They considered 25 primary studies for review purpose and identified several research gap relating to the agile methods, size metrics and cost drivers. Hearty et al. [99] have proposed a Bayesian network model of an XP surrounding and indicated how it could gain from project data keeping in mind the end goal to predict the effort and risk appraisals without obliging any extra metrics.

Popli and Chauhan [100] have proposed a model for effort and cost estimation in agile software development by using regression analysis. Hussain et al. [101] have made an attempt to propose an approach which helps in removing problems like formalized user requirements and thus apply function points for agile software effort estimation. A. E. D. Hamouda [102] have introduced a process and methodology that guarantees relativity in software sizing while using agile story points. This proposed process and methodology was applied in a Capability Maturity Model Integration (CMMI) level three company on different projects. Ungan et al. [103] have compared SCRUM's native effort estimation method Story Points and poker planning, with effort estimation models based on COSMIC Function Points (CFP) for a selection of projects. by using regression models and ANN methodology and proved that COSMIC measurement is a better method for effort estimation than SCRUM's story points.

Viljan Mahnic [104] have described a case study with the aim of studying the behavior of development teams utilizing scrum for the first time, i.e., a situation typical for software companies attempting to bring scrum into their development process. It was found that the initial plans and effort estimates were over-optimistic, but the abilities of estimating and planning improved from sprint to sprint. Mahnic and Zabkar [105] have extended their approach by describing a set of measures that give IT administration with continuous understanding in the scrum-based software development process. The proposed measures were applied within the scope of the project of rebuilding a web site, which served as a contextual investigation for assessment of their ease of use. The contextual investigation demonstrated that each proposed measure depicts a valuable process aspect and collection of data does not require additional administrative work that would harm the agility of scrum. Garg and Gupta [106] have applied Principal Component Analysis (PCA) to reduce the dimensions of the attributes required and identify the key attributes which have maximum correlation to the development cost; and then use constraint solving approach to satisfy the criteria imposed by agile manifesto. The proposed methodology is found to bet suitable for agile projects as it uses constraint programming to explicitly check for satisfaction of agile manifestos. From the analysis

of results, it is found that the proposed model exhibits a low MMRE value than the existing models.

Lenarduzzi et al. [107] have introduced functional size metrics to improve estimation accuracy and to measure the accuracy of expert-based estimation. Further they extended this approach to plain Scrum processes, where the original study was replicated twice, applying an exact replication to two plain Scrum development processes. The results of this replicated study show that the accuracy of the effort estimated by the developers is very accurate and higher than that obtained through functional size measures. Raslan et al. [108] have proposed a framework based on the fuzzy logic which receives fuzzy input parameters of Story Points (SP), Implementation Level Factor (ILF), Friction factors (FR), and Dynamic Forces (DF) to be processed in many successive steps to produce in final the effort estimation. They analyzed the utilization of fuzzy logic in improving the effort estimation accuracy using the user stories by characterizing inputs parameters using trapezoidal membership functions. Britto et al. [109] have performed an empirical investigation on the state of the practice on effort estimation in AGSD. To do so, a survey was carried out using as instrument an on-line questionnaire and a sample comprising software practitioners experienced in effort estimation within the Agile Global Software Development (AGSD) context. Results show that the effort estimation techniques used within the AGSD and collocated contexts remained unchanged, with planning poker being the one employed the most.

2.6 Survey on Software Effort Estimation using Machine Learning Techniques

Alaa Sheta [110] has used Takagi-Sugeno-Kang (TSK) fuzzy model to develop fuzzy models for two different type of nonlinear processes. The first one is based on NASA software projects effort estimation process and the second one is on the stock market prediction process for S & P 500. Kocaguneli et al. [84] have investigates different software effort estimation techniques and found that no single technique reliably beats all others. Subsequently, it is more astute to generate estimates from gatherings of various estimation techniques. Elyassami and Idri [111] have investigate the utilization of Fuzzy choice tree for software effort estimation. The proposed model empower to handle questionable and loose information, which enhance the correctness of obtained evaluations. Pahariya et al. [112] have presented a novel Genetic Algorithm (GA)-based feature selection algorithm for estimating the effort required to develop a software and compared the result with the output obtained using other ML techniques. Results proved that the proposed technique outperformed all

the other existing techniques.

Adriano L.I. Oliveira [12] have provided a comparative study on support vector regression (SVR), radial basis function neural networks (RBFNs) and linear regression for estimation of software project effort. The experiment is carried out using NASA project datasets and the result shows that SVR performs better than RBFN and linear regression. Braga et al. [15] have proposed and investigated the use of a genetic algorithm approach for selecting an optimal feature subset and optimizing SVR parameters simultaneously aiming to improve the precision of the software effort estimates. Kocaguneli et al. [113] have investigated non-uniform weighting through kernel density estimation and found that nonuniform weighting through kernel methods cannot outperform uniform weighting Analogy Based Estimation (ABE). Bakele et al. [114] have proposed ML technique-based effort estimation models and assess the models by taking the help of publicly available resources and data accumulated from software industries. By analyzing the results, it is observed that the utilization of any one model for software effort estimation purpose may not always provide the best possible solution.

Wen et al. [115] have intended to efficiently dissect machine learning models from four viewpoints such as type of machine learning technique, accuracy of estimates, comparison of model, and the context of estimation by taking the help of 84 primary studies. They have observed that the estimation of accuracy obtained using machine learning models is near the satisfactory level and is superior to anything that of non-machine learning models. Azzeh et al. [116] have integrated analogy-based estimation with Fuzzy numbers in order to improve the performance of software project effort estimation during the early stages of a software development life cycle, using all available early data. Results inferred that the proposed similarity measure and adaptation techniques method were able to significantly improve the performance of analogy-based estimation during the early stages of software development and outperform the results of Case-based Reasoning (CBR) and Stepwise Regression.

Azzeh et al. [25] have investigated the potential of ensemble learning for variants of adjustment methods used in analogy-based effort estimation. They performed a large scale comparison study where many ensembles constructed from n out of 40 possible valid variants of adjustment methods are applied to eight datasets. The results have been subjected to statistical significance testing, and show reasonable significant improvements on the predictive performance where ensemble methods are applied. Mendes et al. [117] have described an industrial case study where an expert-based requirements effort estimation model was built and validated for the Brazilian Navy. A knowledge engineering of Bayesian networks process was employed to build the requirements effort estimation model. The expert-based requirements

effort estimation model was built with the participation of seven software requirements analysts and project managers, leading to 28 number of prediction factors and more than 30 number of relationships. The model was validated based on real data from 11 number of large requirements specification. The model was incorporated into the Brazilian navys quality assurance process to be used by their software requirements analysts and managers.

2.7 Summary of Observations

For effort estimation process, it can be observed that function point and COCOMO has been used by a number of researchers as input in design of prediction models. It is observed that the public datasets are also most commonly used in object oriented software effort estimation. Traditional software estimation techniques like Constructive Cost Estimation Model (COCOMO) and Function Point Analysis (FPA) have been proved to be unsatisfactory for measuring the cost and effort of all types of software development. This is because the line of code (LOC) and function point (FP) were both used for procedural programming concept. The procedural oriented design splits the data and procedure, whereas the object oriented design combines both of them. But, as Unified Modeling Language (UML) diagrams become a popular approach to represent object-oriented softwares, the use of class point approach derived from Class Diagram and Use Case point approach based on the requirement analysis phases as well as from use case diagram are the solutions, which will have wider acceptance for object-oriented software effort estimation purpose.

With the rising use of dependency on Web, there is a necessity of quick and efficient development of web-based software. For developing web-based software efficiently i.e. without any cost or resource (human or otherwise) overrun, the estimates that are done before the beginning of development need to be correct. It is noticed that many authors have used Tukutuku dataset for web effort estimation process, which is not publicly available. Very few authors have used the ISBSG dataset for web effort estimation. Hence, by more exploring the application of ISBSG dataset for web effort estimation process and improving the prediction accuracy by using statistical and machine learning techniques, it will provide more flexibility as well as better prediction accuracy for the industries to estimate the effort of web applications development.

Agile software effort estimation is also one of the promising area of research. Many researchers have proposed various methodologies for agile software development process. But there is lack of availability of good amount of research work for providing a systematic procedure in order to estimate the effort of softwares developed using agile methodology. Story point approach is one of the popular ways to estimate the effort of

softwares developed using scrum methodology. But there is a very scarcity of dataset based on story point approach due to unavailability of project velocity information. Hence, by applying the different statistical and machine learning techniques on the considered story point dataset, the accuracy of agile software effort estimation process will be improved.

Chapter 3

Class Point Approach for Software Effort Estimation using Machine Learning Techniques

3.1 Introduction

Object-oriented (OO) technology is the accepted methodology in the present day scenario for software development in major industries as it helps in building software development process in a more organized fashion [118]. With the increase in the complexities associated with modern day software projects, the need for early and accurate effort estimation in the software development phase has become pivotal. Currently used effort estimation techniques like Function Point Approach (FPA) and COCOMO, can not be claimed as the most efficient techniques to estimate the cost and effort required to develop the software [119, 120]. These techniques are not capable of measuring efficiently the cost and effort, because they are tailored for procedural-oriented software development paradigm [121]. The procedural oriented paradigm and object-oriented paradigm differ because the former splits the data and procedure; while the later combines them.

It is important to realize that the problem of learning/estimation of dependencies from samples is only one part of the general experimental procedure used by researchers and practitioners who apply statistical methods to draw conclusions from the data [122, 123]. Hence to obtain proper results in estimating software size, it is essential to consider the data obtained from previous projects. As far as effort estimation is concerned, a number of unsolved problems and errors still exist. Estimation of a software project is always important aspect for determining the feasibility of the project [124]. In the present scenario, most of the software project planning activities depend upon estimated figures of effort. Since *class* is the fundamental logical unit of an OO system, the utilization of the class point

methodology to compute the project effort serves as a basic guideline. During the calculation procedure of the final adjusted class point, two measures, Class Point 1 (CP1) and Class Point 2 (CP2), are utilized. CP1 is figured utilizing two measures, Number of External Methods (NEM) and Number of Services Requested (NSR); whereas CP2 is ascertained by utilizing an alternate metric as a part of expansion to NEM and NSR, Number of Attributes (NOA). NEM figures the measure of the interface of a class and is directed by the measure of local public methods, although NSR gives a measure of the linkage of the components of the software system. On the other hand, NOA helps in finding out the number of attributes utilized in a class. In case of FPA and CPA, the Technical Complexity Factor (TCF) is calculated based on the impact of various general characteristics of a system. However, in both these cases, the non-technical factors such as effectiveness of the management, technical competence of developers, security of the system, system's reliability, system's maintenance capability and system's portability are not looked into [54]. Hence in this study, the optimized CPA is utilized to ascertain the effort needed to create the software adopting these six non-technical factors. Likewise with a specific end goal to accomplish an improved value of prediction accuracy, Stochastic Gradient Boosting (SGB) and four Support Vector Regression (SVR) Kernels-based effort estimation models are applied over the obtained class point value. The results obtained from the these models are then compared with the results obtained from other machine learning techniques available in literature in order to access their performance.

3.2 Methodologies Used

The following methodologies are used in this research to calculate the effort of a software product.

3.2.1 Class Point Approach (CPA)

The CPA was presented by Costagliola et al. [125] in 1998. It is focused around the FPA methodology to speak to the interior qualities of a software. The essential thought of the CPA system is calculation of number of classes in a project. It is derived from the perception that in the procedural model, functions or methods are the essential programming units; while, in the OO model, classes are the coherent building pieces. The block diagram, demonstrated in figure 3.1, displays the steps to compute the project development effort using class point approach [126–130].

The system to acquire the amount of class points is isolated into three principal

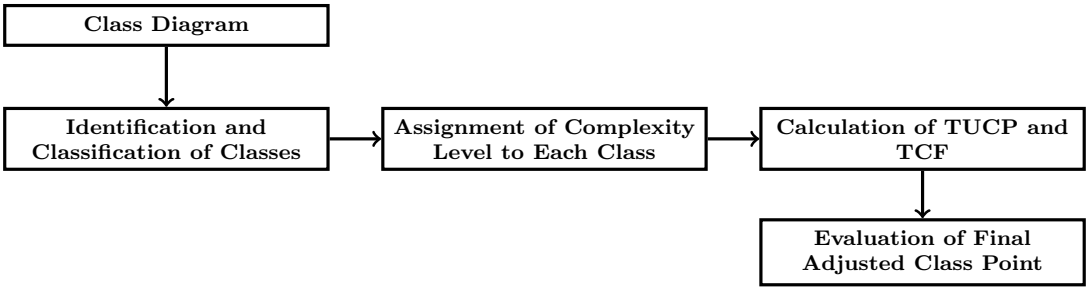


Figure 3.1: Steps to Calculate Final Adjusted Class Point

stages [53].

- Estimation of information processing size
  - Identification and classification of classes
  - Evaluation of complexity level for each classified class
  - Calculation of the Total Unadjusted Class Points (TUCP) value
- Estimation of the Technical Complexity Factor (TCF) value
- Calculation of the final value of Adjusted Class Point (ACP)

Identification and Classification of Classes

During the first step, the design specifications are analyzed in order to identify and classify the classes into four types of system components, namely the Problem Domain Type (PDT), the Human Interaction Type (HIT), the Data Management Type (DMT), and the Task Management Type (TMT) [53]. The PDT component contains classes representing real-world entities in the application domain of the system. The classes of HIT type are designed to satisfy the need for information visualization and human-computer interaction. The DMT component encompasses the classes that offer functionality for data storage and retrieval. Finally, TMT classes are designed for task management purposes, thus they are responsible for the definition and control of tasks.

Evaluation of complexity level for each classified class

During the second step, each identified class is assigned a complexity level, which is determined on the basis of methods associated with the class and of the interaction of the class with the rest of the system. In some cases, the complexity level of each class is determined on the basis of the NEM, and NSR. In some other cases, besides the above measures, the NOA measure is taken into account in order to evaluate the

complexity level of each class. For the calculation of CP1, the complexity level of the class is determined based on the value of NEM and NSR according to Table 3.1. For example, if a class is having NEM value 7 and NSR value 3, then the complexity level can be assigned to the class as 'Average'.

Table 3.1: Complexity Level Evaluation for CP1

	0 - 4 NEM	**5 - 8 NEM**	**9 - 12 NEM**	$\geq$ 13 **NEM**
0 - 1 NSR	Low	Low	Average	High
2 - 3 NSR	Low	Average	High	High
4 - 5 NSR	Average	High	High	Very High
$>$ **5 NSR**	High	High	Very High	Very High

For the calculation of CP2, the complexity level of the class is determined based on the value of NEM, NOA and NSR according to Table 3.2. From Table 3.2, it is observed that ranges of NEM and NOA vary with respect to fixed NSR range.

Table 3.2: Complexity Level Evaluation for CP2

0 - 2 NSR	**0 - 5 NOA**	**6 - 9 NOA**	**10 - 14 NOA**	$\geq$ 15 **NOA**
0 - 4 NEM	Low	Low	Average	High
5 - 8 NEM	Low	Average	High	High
9 - 12 NEM	Average	High	High	Very High
$\geq$ 13 **NEM**	High	High	Very High	Very High

(a)

3 - 4 NSR	**0 - 4 NOA**	**5 - 8 NOA**	**9 - 13 NOA**	$\geq$ 14 **NOA**
0 - 3 NEM	Low	Low	Average	High
4 - 7 NEM	Low	Average	High	High
8 - 11 NEM	Average	High	High	Very High
$\geq$ 12 **NEM**	High	High	Very High	Very High

(b)

$\geq$ 5 ***NSR***	**0 - 3 NOA**	**4 - 7 NOA**	**8 - 12 NOA**	$\geq$ 13 **NOA**
0 - 2 NEM	Low	Low	Average	High
3 - 6 NEM	Low	Average	High	High
7 - 10 NEM	Average	High	High	Very High
$\geq$ 11 **NEM**	High	High	Very High	Very High

(c)

Calculation of the Total Unadjusted Class Points (TUCP) value

Once a complexity level of each class has been assigned, such information and its type are used to assign a weight to the class given in Table 3.3. Then, the Total Unadjusted

Class Point value (TUCP) is computed as a weighted sum as shown below:

$$TUCP = \sum_{i=1}^{4} \sum_{j=1}^{3} w_{ij} \times x_{ij} \tag{3.1}$$

where x_{ij} is the number of classes of component type i (problem domain, human interaction, etc.) with complexity level j (low, average, or high), and w_{ij} is the weighting value for type i and complexity level j.

Table 3.3: Evaluation of TUCP Value for Each Class Type

System Component Type	Description	Complexity			
		Low	Average	High	Very High
PDT	Problem Domain Type	3	6	10	15
HIT	Human Interaction Type	4	7	12	19
DMT	Data Management Type	5	8	13	20
TMT	Task Management Type	4	6	9	13

Estimation of the Technical Complexity Factor (TCF) value

The Technical Complexity Factor (TCF) is determined by adjusting the TUCP with a value obtained by 24 different target software system characteristics, each on a scale of 0 to 5. The sum of the influence degrees related to such general system characteristics forms the Total Degree of Influence (TDI) as shown in Table 3.4, which is used to determine the TCF according to the following formula:

$$TCF = 0.55 + (0.01 * TDI) \tag{3.2}$$

Out of all the twenty-four characteristics, some of them are very important for object-oriented systems such as user adaptivity, rapid prototyping, multiuser interactivity, multiple interface. Two characteristics i.e., management efficiency and developers' professional competence help for calculation of CP2. These two are not considered for CP1 calculation. The complex processing characteristic describes the complexity level of the tasks. Similarly, developers' professional competence characteristic describes the technology and skills, the developer of project possess. If the developers' have good professional competence, then it will be easier for them to develop the project effectively. The facilitation of change characteristic denotes the ability to adopt changes quickly; whereas maintainability characteristic denotes a number of aspects such as isolation, correction and prevention of defects, maximization of products life cycle, maximization of efficiency, reliability and security of software project etc.

Table 3.4: Degree of Influences of 24 General System Characteristics

ID	*System Characteristics*	*DI*	*ID*	*System Characteristics*	*DI*
C1	Data Communication	....	C13	Multiple sites	....
C2	Distributed Functions	....	C14	Facilitation of change	....
C3	Performance	....	C15	User Adaptivity	....
C4	Heavily used configuration	....	C16	Rapid Prototyping	....
C5	Transaction rate	....	C17	Multiuser Interactivity	....
C6	Online data entry	....	C18	Multiple Interfaces	....
C7	End-user efficiency	....	C19	Management Efficiency	....
C8	Online update	....	C20	Developers' Professional Competence	....
C9	Complex processing	....	C21	Security	....
C10	Reusability	....	C22	Reliability	....
C11	Installation ease	....	C23	Maintainability	....
C12	Operational ease	....	C24	Portability	....
TDI				Total Degree of Influence (TDI)	....

Calculation of the final value of Adjusted Class Point (ACP)

Finally, the Adjusted Class Point (ACP) value is determined by multiplying the Total Unadjusted Class Point (TUCP) value by TCF.

$$ACP = TUCP * TCF \tag{3.3}$$

In this study, the above phases are followed to calculate final optimized class points. Herein, the total number of class point value is then used as an input parameter to the ML techniques-based effort estimations models to calculate the estimated effort.

3.3 Proposed Approach

The proposed approach is tested by using dataset that contains data derived from forty projects [53], which were developed using the Java language. The dataset is used to evaluate software development effort and to validate the improvement. The results obtained in the validation process provided initial experimental evidence of the effectiveness of CPA. The dataset is used to develop the SGB and different SVR kernel-based software effort estimation model. The block diagram, shown in figure 3.2, displays the proposed steps used to determine the predicted effort using the SGB and four SVR kernel techniques. To calculate the effort of a given software project, the following steps are used.

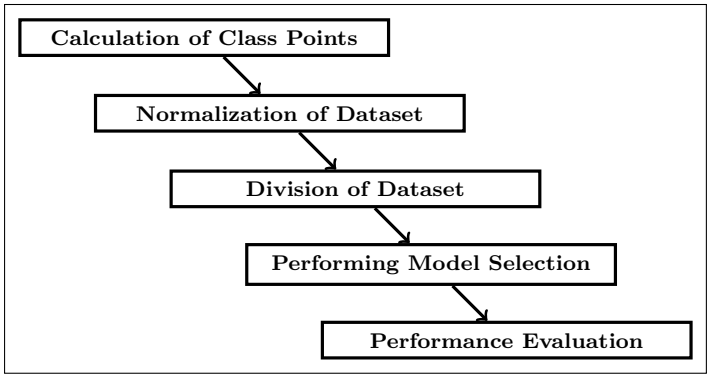


Figure 3.2: Proposed Steps Used for the Effort Estimation based on CPA using SGB and SVR Kernel Techniques

Steps in Effort Estimation

1. **Calculation of Class Points**: In this step, the CP1 and CP2 values are calculated from the class diagram. Generated CP1 and CP2 values are used as an input argument.

2. **Normalization of Dataset**: Input parameter values are individually normalized to the range [0,1]. Let X be the dataset and x be an element of the dataset. Then, the normalized value of x can be calculated as :

$$Normalized(x) = \frac{x - \min(X)}{\max(X) - \min(X)} \tag{3.4}$$

   where
   $\min(X)$ represents the minimum value of the dataset X and $\max(X)$ represents the maximum value of the dataset X. If $\max(X)$ is equal to $\min(X)$, then Normalized(x) is set to 0.5.

3. **Division of Dataset**: The dataset is divided into three parts, such as learning set, validation set and test set.

4. **Performing Model Selection**: To select effort estimation model based on SGB technique, first the values of various parameters such as number of trees, Huber's quantile cutoff, shrinkage factor, stochastic factor and influence trimming factor are found out and then a five-fold cross validation is implemented for model selection. The advantage of using cross validation is to avoid the possible biasness introduced by relying on any particular division of dataset for training and testing. It provides the advantage of using all data for training and testing purpose. The model that provides the lowest RMSE,

MAE, MMRE, MMER values and the highest prediction accuracy (PRED (x)) values is selected as the best model for each fold.

Similarly, in case of SVR kernel-based effort estimation model, the model which provides the lesser value than the other generated models based on the minimum validation error criteria has been selected to perform other operations. The tunable parameters are selected to find the most suitable parameters of C and γ using a five-fold cross validation procedure . Based on the minimum validation error, the best model has been selected and the corresponding values of γ as well as ϵ are found out. The final model selected based on best parameter of C, ϵ and γ have been trained using all training samples. The output of this step is the trained SVM model providing predicted response values for test inputs.

5. **Performance Evaluation**: The performance of the model is verified using final RMSE, MAE, MMRE, MMER and PRED(x) values obtained from test samples. The obtained values of various models are compared with the existing results as well as among themselves in order to access their accuracy.

The above steps are followed to implement the SGB and SVR Kernel-based effort estimation models. Finally, a comparison of the results obtained using these models with the results obtained from the other models is presented with an objective to assess their performances.

3.4 Experimental Details

In the proposed research study, the dataset collected from Costagliola et al. [53] and Zhou and Liu [54], listed in Tables 3.5 and 3.6 respectively, are used. In these tables, every row displays the details of one project developed in the JAVA language values of CP1, CP2 and the actual effort (denoted by EFH) expressed in terms of person-hours required to successfully complete the project.

The statistical profile of two different categories of dataset collected for Class Point Approach is depicted in Table 3.7. From this table, it can be observed that both the 40 and 30 projects datasets are more normally distributed based on the values of the skewness and kurtosis.

Figures 3.3a and 3.3b depict the relationship between software size and software effort (person-hours) based on CP1 & CP2, using the dataset of 40 projects. Similarly, Figures 3.3c and 3.3d depict the relationship between software size and software effort (person-hours) based on CP1 & CP2 using 30 project dataset respectively.

From these figures, it is observed that the 30 project dataset contains more number of outliers than 40 project dataset. Figures 3.4a and 3.4b display the histogram of

Table 3.5: Forty Project Dataset [53]

	EFH	CP1	CP2
1	286	103.18	110.55
2	396	278.72	242.54
3	471	473.90	446.60
4	1016	851.44	760.96
5	1261	1263.12	1242.60
6	261	196.68	180.84
7	993	178.80	645.60
8	552	213.30	208.56
9	998	1095.00	905.00
10	180	116.62	95.06
11	482	267.80	251.55
12	1083	687.57	766.29
13	205	59.64	64.61
14	851	697.48	620.10
15	840	864.27	743.49
16	1414	1386.32	1345.40
17	279	132.54	74.26
18	621	550.55	481.66
19	601	539.35	474.95
20	680	489.06	438.90
21	366	287.97	262.74
22	947	663.60	627.60
23	485	397.10	358.60
24	812	678.28	590.42
25	685	386.31	428.18
26	638	268.45	280.84
27	1803	2090.70	1719.25
28	369	114.40	104.50
29	439	162.87	156.64
30	491	258.72	246.96
31	484	289.68	241.40
32	481	480.25	413.10
33	861	778.75	738.70
34	417	263.72	234.08
35	268	217.36	195.36
36	470	295.26	263.07
37	436	117.48	126.38
38	428	146.97	148.35
39	436	169.74	200.10
40	356	112.53	110.67

Table 3.6: Thirty Project Dataset [54]

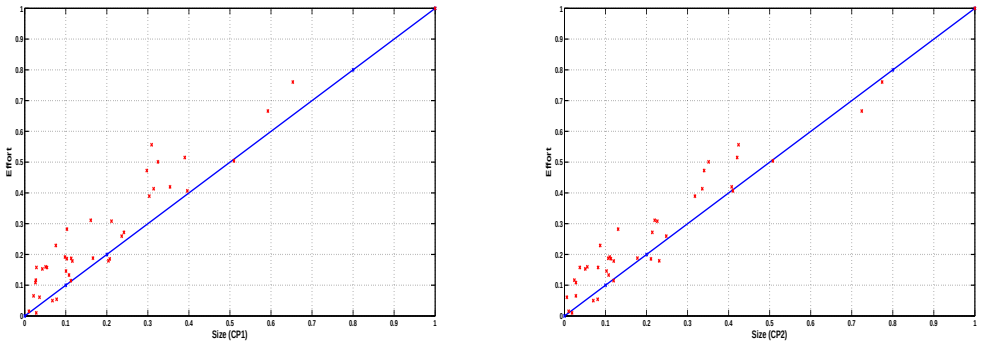
	EFH	CP1	CP2
1	286	103.18	110.55
2	396	278.72	242.54
3	471	473.90	446.60
4	1016	851.44	760.96
5	1261	1263.12	1242.60
6	261	196.68	180.84
7	993	178.80	645.60
8	552	213.30	208.56
9	998	1095.00	905.00
10	180	116.62	95.06
11	482	267.80	251.55
12	1083	687.57	766.29
13	205	59.64	64.61
14	851	697.48	620.10
15	840	864.27	743.49
16	1414	1386.32	1345.40
17	279	132.54	74.26
18	621	550.55	481.66
19	601	539.35	474.95
20	680	489.06	438.90
21	366	287.97	262.74
22	947	663.60	627.60
23	485	397.10	358.60
24	812	678.28	590.42
25	685	386.31	428.18
26	638	268.45	280.84
27	1803	2090.70	1719.25
28	369	114.40	104.50
29	439	162.87	156.64
30	491	258.72	246.96

Table 3.7: Statistical Profile of Two Datasets used for Class Point Approach

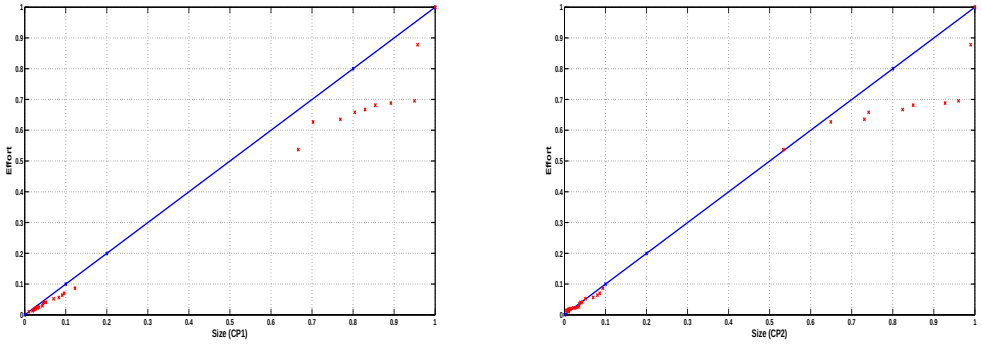
Project Type	Minimum	Maximum	Mean	Median	Std. Dev.	Skewness	Kurtosis
40 Project Dataset	180	1803	628.55	484.5	351.43	1.37	2.11
30 Project Dataset	55	2895	787.87	187	941.97	0.92	-0.86

effort value for 40 and 30 project dataset respectively. From these figures, it can been observed that 40 project dataset are more normally distributed based on the values of the skewness and kurtosis than 30 project dataset, which can also be verified based on values of skewness and kurtosis provided in Table 3.7,

After calculating the final class point values, the dataset is then normalized.

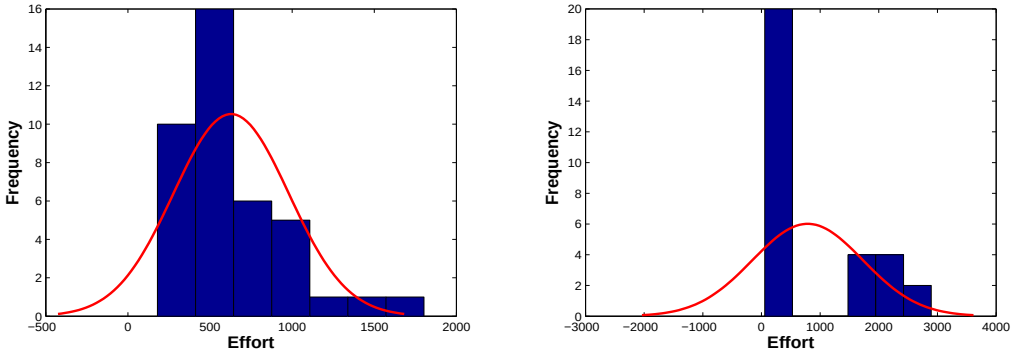


(a) (b)



(c) (d)

Figure 3.3: Software Size vs. Effort Graph based on CP1 & CP2 using 40 and 30 Project Datasets



(a) (b)

Figure 3.4: Histogram of Effort Values for 40 and 30 Project Dataset

The normalized dataset is divided into different subsets using a double sampling procedure. In the sampling procedure, the normalized dataset is first divided into a *training set* and a *test set*. The *training set* is used for learning purposes (model estimation), whereas the *test set* is used only for estimating the prediction accuracy

of the final model. In the second step, the *training set* is divided into a *learning set* and a *validation set*. The *learning set* is used to estimate model parameters, and the *validation set* is used for selecting an optimal model (usually via cross validation).

Every fifth row of the Tables 3.5 and 3.6, is extracted for testing purposes and the rest are used for training purposes. Hence, after completion of the first step of the double sampling process, the complete forty project dataset is divided into thirty-two rows for training and eight rows for testing. Then, every fifth row of the training set is extracted for validation purposes, and the rest are used for learning purposes. Hence, after completion of the second step of the double sampling process, the complete thirty-two project dataset is divided into twenty-six rows for learning and six rows for validation. After partitioning the sample into a learning set and a validation set, the model selection is performed using a five fold cross validation process.

3.4.1 Model Design using Stochastic Gradient Boosting Technique

To design an effort estimation model using the SGB technique, the following steps are used.

1. The coefficient of F0 is obtained by calculating the mean of the target variables (Software Effort).

2. A random percentage of rows are selected to feed the next tree using the stochastic factor. If it is set to 0.5, 50 percent of the rows will be randomly chosen.

3. The residuals of the rows are sorted and then the residues using the Huber's Quantile Cutoff factor are transfered. The transformed residual values are called *pseudo-residuals*.

4. The first tree (T1) is fitted to the pseudo-residuals.

5. The predicted values of the nodes are calculated using the mean of the pseudo-residuals in each of the terminal nodes.

6. The residuals between the predicted values and the pseudo-residuals that fed the tree are calculated.

7. The Huber's Quantile Cutoff factor is applied again on the result obtained from step 6 and the mean of these residuals are then computed.

8. The boost coefficient (A1) of the tree is obtained by measuring the difference between the mean residual value and the mean of the predicted values of the tree.

9. Finally, the boost coefficient is multiplied by the shrink value to retard the learning process.

The following parameter values are chosen to predict the effort using the SGB technique.

- No of Trees : 1000
- Huber's Quantile Cut off : 0.95
- Shrinkage Factor : 0.1
- Stochastic Factor : 0.5
- Influence Trimming Factor : 0.01

The detailed descriptions of these parameters were already provided in Section 1.4.2. The values for the parameters are assigned by choosing appropriate combinations of values to generate the best result for the SGB-based effort estimation model.

Figures 3.5a and 3.5b display the actual effort and the predicted effort obtained for CP1 and CP2 respectively using the SGB technique, taking into consideration of 40 project dataset. Similarly, Figures 3.5c and 3.5d display the actual effort and the predicted effort obtained for CP1 and CP2 respectively using the SGB technique taking into consideration of 30 project dataset. From this figure, it is observed that there is very little difference between the predicted effort and the actual effort.

3.4.2 Model Design using Various SVR Kernel Methods

After partitioning data into learning set and validation set, the model selection for ϵ and γ is performed using 5-fold cross validation process. In this study, in order to perform model selection, the ϵ and γ values are varied over a range. The γ value ranges from 2^{-7} to 2^{7} and ϵ value ranges from 0 to 5. Hence, ninety models are generated to perform model selection operation.

Tables 3.8 and 3.9 show the validation error of ninety numbers of models generated for CP1 using SVR linear kernel and SVR polynomial kernel respectively based on the values of ϵ and γ for 40 project dataset. For SVR Linear kernel, *0.0065* value has been chosen as the minimum validation error. Hence based on the minimum

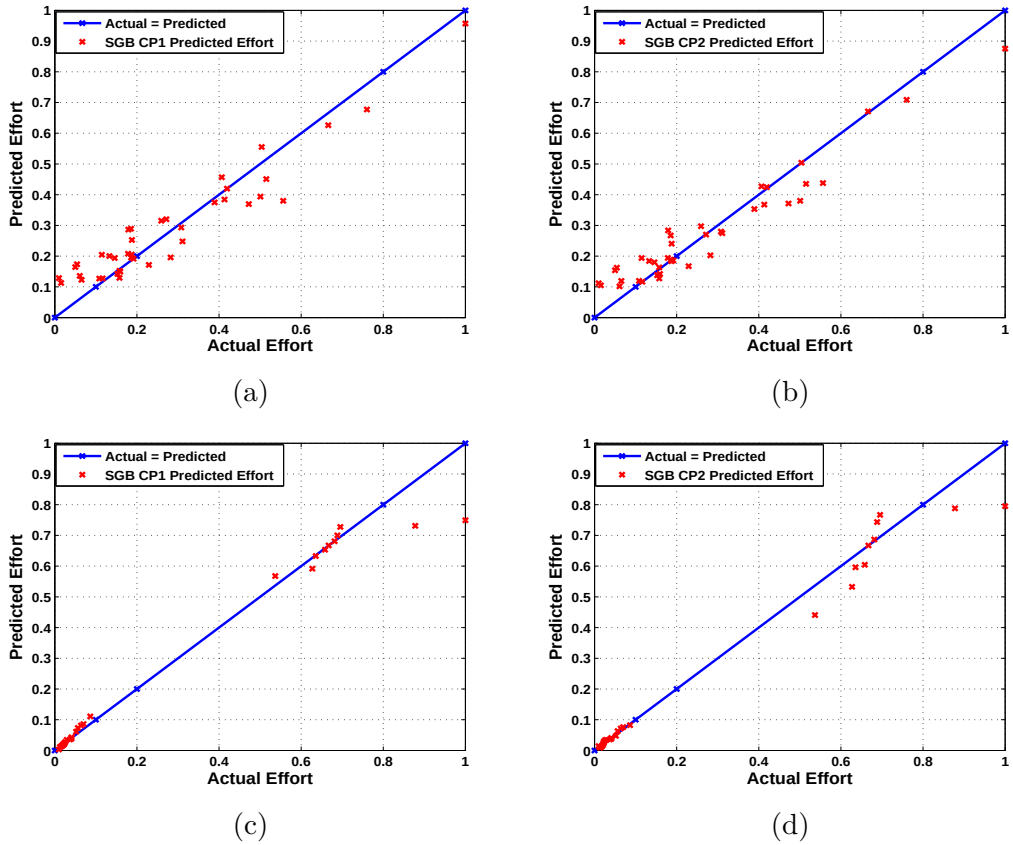


Figure 3.5: Actual vs. Predicted Effort Graph using the SGB Technique for 40 and 30 Project Datasets

Table 3.8: Validation Errors Obtained Using SVR Linear Kernel for CP1

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	**0.0065**	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-6}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-5}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-4}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-3}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-2}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-1}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177
2^{0}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177
2^{1}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177
2^{2}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177
2^{3}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177
2^{4}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177
2^{5}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177
2^{6}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177
2^{7}	0.0065	0.1177	0.1177	0.1177	0.1177	0.1177

Table 3.9: Validation Errors Obtained Using SVR Polynomial Kernel for CP1

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.0445	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-6}	0.0445	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-5}	0.0445	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-4}	0.0445	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-3}	0.0443	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-2}	0.0432	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-1}	0.0348	0.1177	0.1177	0.1177	0.1177	0.1177
2^{0}	**0.0128**	0.1177	0.1177	0.1177	0.1177	0.1177
2^{1}	0.0318	0.1177	0.1177	0.1177	0.1177	0.1177
2^{2}	0.0402	0.1177	0.1177	0.1177	0.1177	0.1177
2^{3}	0.0402	0.1177	0.1177	0.1177	0.1177	0.1177
2^{4}	0.0402	0.1177	0.1177	0.1177	0.1177	0.1177
2^{5}	0.0402	0.1177	0.1177	0.1177	0.1177	0.1177
2^{6}	0.0400	0.1177	0.1177	0.1177	0.1177	0.1177
2^{7}	0.0389	0.1177	0.1177	0.1177	0.1177	0.1177

validation error, the best model is C = 0.99, γ = 0.0078125 and ϵ = 0. For SVR Polynomial kernel, *0.0128* value has been chosen as the minimum validation error. Based on the minimum validation error, the best model is C = 0.99, γ = 1 and ϵ = 0.

Tables 3.10 and 3.11 show the validation error of ninety numbers of models generated for CP1 using SVR RBF kernel and SVR Sigmoid kernel respectively based

Table 3.10: Validation Errors Obtained Using SVR RBF Kernel for CP1

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.0396	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-6}	0.0351	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-5}	0.0272	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-4}	0.0162	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-3}	0.0107	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-2}	0.0087	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-1}	**0.0073**	0.1177	0.1177	0.1177	0.1177	0.1177
2^{0}	0.0085	0.1177	0.1177	0.1177	0.1177	0.1177
2^{1}	0.0126	0.1177	0.1177	0.1177	0.1177	0.1177
2^{2}	0.0126	0.1177	0.1177	0.1177	0.1177	0.1177
2^{3}	0.0099	0.1177	0.1177	0.1177	0.1177	0.1177
2^{4}	0.0138	0.1177	0.1177	0.1177	0.1177	0.1177
2^{5}	0.0193	0.1177	0.1177	0.1177	0.1177	0.1177
2^{6}	0.0213	0.1177	0.1177	0.1177	0.1177	0.1177
2^{7}	0.0234	0.1177	0.1177	0.1177	0.1177	0.1177

Table 3.11: Validation Errors Obtained Using SVR Sigmoid Kernel for CP1

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.0420	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-6}	0.0396	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-5}	0.0351	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-4}	0.0271	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-3}	0.0159	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-2}	0.0102	0.1177	0.1177	0.1177	0.1177	0.1177
2^{-1}	**0.0080**	0.1177	0.1177	0.1177	0.1177	0.1177
2^{0}	0.0083	0.1177	0.1177	0.1177	0.1177	0.1177
2^{1}	0.0159	0.1177	0.1177	0.1177	0.1177	0.1177
2^{2}	0.1894	0.1177	0.1177	0.1177	0.1177	0.1177
2^{3}	1.3186	0.1177	0.1177	0.1177	0.1177	0.1177
2^{4}	3.0569	0.1177	0.1177	0.1177	0.1177	0.1177
2^{5}	3.5491	0.1177	0.1177	0.1177	0.1177	0.1177
2^{6}	2.9537	0.1177	0.1177	0.1177	0.1177	0.1177
2^{7}	2.3651	0.1177	0.1177	0.1177	0.1177	0.1177

on the values of ϵ and γ for 40 project dataset. For SVR RBF kernel, *0.0073* value has been chosen as the minimum validation error. Based on the minimum validation error, the best model is C = 0.99, $\gamma = 0.5$ and $\epsilon = 0$. For SVR Sigmoid kernel, *0.0080* value has been chosen as the minimum validation error. Based on the minimum validation error, the best model is C = 0.99, $\gamma = 0.5$ and $\epsilon = 0$. Based on model parameters' value, the model has been trained and tested using training and testing data set respectively to estimate the effort.

Table 3.12: Validation Errors Obtained Using SVR Linear Kernel for CP2

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	**0.0055**	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-6}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-5}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-4}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-3}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-2}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-1}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135
2^{0}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135
2^{1}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135
2^{2}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135
2^{3}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135
2^{4}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135
2^{5}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135
2^{6}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135
2^{7}	0.0055	0.1135	0.1135	0.1135	0.1135	0.1135

Table 3.13: Validation Errors Obtained Using SVR Polynomial Kernel for CP2

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.0484	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-6}	0.0484	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-5}	0.0484	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-4}	0.0484	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-3}	0.0483	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-2}	0.0470	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-1}	0.0380	0.1135	0.1135	0.1135	0.1135	0.1135
2^{0}	**0.0186**	0.1135	0.1135	0.1135	0.1135	0.1135
2^{1}	0.0583	0.1135	0.1135	0.1135	0.1135	0.1135
2^{2}	0.0572	0.1135	0.1135	0.1135	0.1135	0.1135
2^{3}	0.0572	0.1135	0.1135	0.1135	0.1135	0.1135
2^{4}	0.0572	0.1135	0.1135	0.1135	0.1135	0.1135
2^{5}	0.0573	0.1135	0.1135	0.1135	0.1135	0.1135
2^{6}	0.0572	0.1135	0.1135	0.1135	0.1135	0.1135
2^{7}	0.0575	0.1135	0.1135	0.1135	0.1135	0.1135

Tables 3.12 and 3.13 show the validation error of ninety numbers of models generated for CP2 using SVR linear kernel and SVR polynomial kernel respectively based on the values of ϵ and γ for 40 project dataset. For SVR Linear kernel, *0.0055* value has been chosen as the minimum validation error. Hence based on the minimum validation error, the best model is C = 0.99, $\gamma = 0.0078125$ and $\epsilon = 0$. For SVR Polynomial kernel, *0.0186* value has been chosen as the minimum validation error. Hence based on the minimum validation error, the best model is C = 0.99, $\gamma = 1$ and

$\epsilon = 0$.

Table 3.14: Validation Errors Obtained Using SVR RBF Kernel for CP2

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.0429	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-6}	0.0381	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-5}	0.0299	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-4}	0.0178	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-3}	0.0094	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-2}	0.0070	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-1}	0.0065	0.1135	0.1135	0.1135	0.1135	0.1135
2^{0}	**0.0063**	0.1135	0.1135	0.1135	0.1135	0.1135
2^{1}	0.0096	0.1135	0.1135	0.1135	0.1135	0.1135
2^{2}	0.0110	0.1135	0.1135	0.1135	0.1135	0.1135
2^{3}	0.0090	0.1135	0.1135	0.1135	0.1135	0.1135
2^{4}	0.0111	0.1135	0.1135	0.1135	0.1135	0.1135
2^{5}	0.0178	0.1135	0.1135	0.1135	0.1135	0.1135
2^{6}	0.0199	0.1135	0.1135	0.1135	0.1135	0.1135
2^{7}	0.0226	0.1135	0.1135	0.1135	0.1135	0.1135

Table 3.15: Validation Errors Obtained Using SVR Sigmoid Kernel for CP2

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.0455	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-6}	0.0429	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-5}	0.0380	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-4}	0.0298	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-3}	0.0174	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-2}	0.0091	0.1135	0.1135	0.1135	0.1135	0.1135
2^{-1}	**0.0066**	0.1135	0.1135	0.1135	0.1135	0.1135
2^{0}	0.0076	0.1135	0.1135	0.1135	0.1135	0.1135
2^{1}	0.0206	0.1135	0.1135	0.1135	0.1135	0.1135
2^{2}	0.1967	0.1135	0.1135	0.1135	0.1135	0.1135
2^{3}	1.3430	0.1135	0.1135	0.1135	0.1135	0.1135
2^{4}	3.1364	0.1135	0.1135	0.1135	0.1135	0.1135
2^{5}	3.6039	0.1135	0.1135	0.1135	0.1135	0.1135
2^{6}	3.0072	0.1135	0.1135	0.1135	0.1135	0.1135
2^{7}	2.3954	0.1135	0.1135	0.1135	0.1135	0.1135

Tables 3.14 and 3.15 show the validation error of ninety numbers of models generated for CP2 using SVR RBF kernel and SVR Sigmoid kernel respectively based on the values of ϵ and γ for 40 project dataset. For SVR RBF kernel, *0.0063* value has been chosen as the minimum validation error. Based on the minimum validation error, the best model is C = 0.99, $\gamma = 1$ and $\epsilon = 0$. For SVR Sigmoid kernel, *0.0066* value has been chosen as the minimum validation error. Based on the minimum validation error, the best model is C = 0.99, $\gamma = 0.5$ and $\epsilon = 0$. Based on model parameters' value, the model has been trained and tested using training and testing data set respectively to estimate the effort.

While using the RMSE, MMRE and PRED in evaluation, convincingly better results are implied by lower values of RMSE, MMRE and higher value of PRED. After implementing the support vector regression based model using four different kernel methods for software effort estimation using 40 project dataset, the following results have been generated.

SVR Linear Kernel Result for CP1:

Param: -s 3 -t 0 -c 0.99 -g 0.0078125 -p 0

* Mean Squared Error (MSE_TEST) = 0.0047

* Squared correlation coefficient = 0.9065

SVR Polynomial Kernel Result for CP1:

Param: -s 3 -t 1 -c 0.99 -g 1 -p 0

* Mean Squared Error (MSE_TEST) = 0.0234

* Squared correlation coefficient = 0.5458

SVR RBF Kernel Result for CP1:

Param: -s 3 -t 2 -c 0.99 -g 0.5 -p 0

* Mean Squared Error (MSE_TEST) = 0.0036

* Squared correlation coefficient = 0.9298

SVR Sigmoid Kernel Result for CP1:

Param: -s 3 -t 3 -c 0.99 -g 0.5 -p 0

* Mean Squared Error (MSE_TEST) = 0.0050

* Squared correlation coefficient = 0.9092

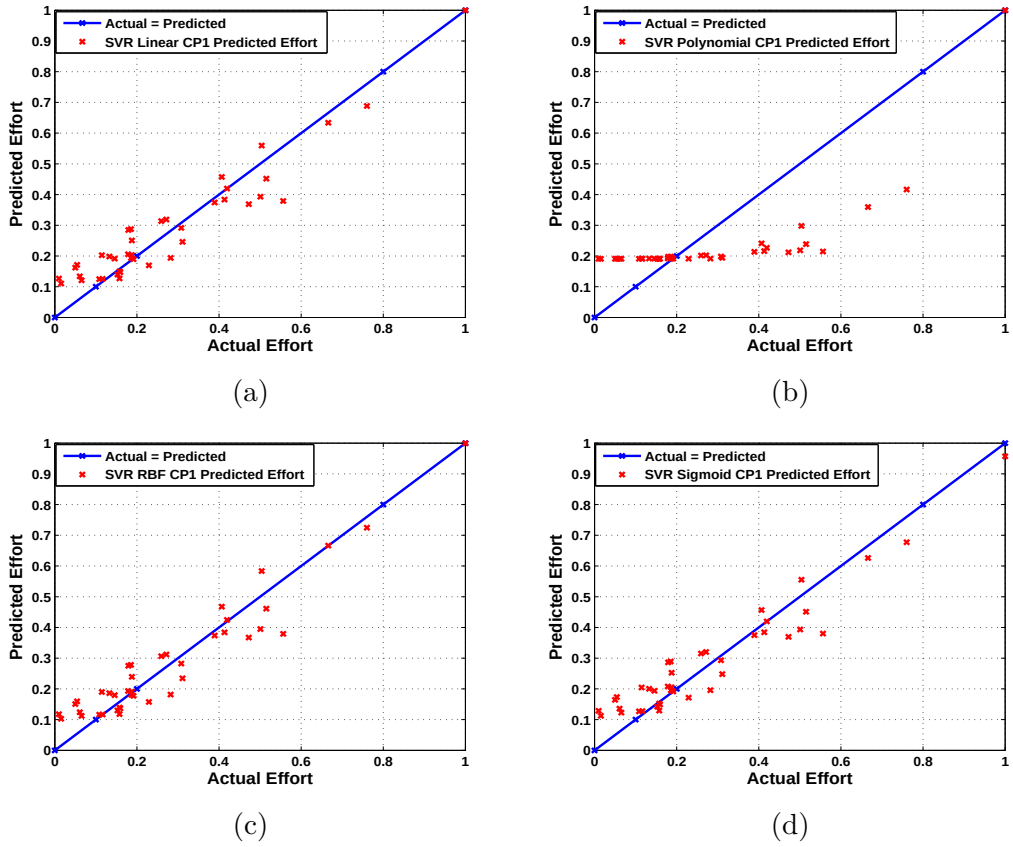


Figure 3.6: SVR Linear, Polynomial, RBF and Sigmoid Kernel based Effort Estimation Model for CP1 using 40 Project Dataset

The proposed model generated values for CP1 using the SVR linear, polynomial, RBF and sigmoid kernel have been plotted as shown in Figures 3.6a, 3.6b, 3.6c and 3.6d respectively. These figures display the variation of actual effort and the predicted effort obtained for CP1 using the four SVR kernel methods taking into consideration of 40 project dataset.

SVR Linear Kernel Result for CP2:

Param: -s 3 -t 0 -c 0.99 -g 0.0078125 -p 0

* Mean Squared Error (MSE_TEST) = 0.0033

* Squared correlation coefficient = 0.9341

SVR Polynomial Kernel Result for CP2:

Param: -s 3 -t 1 -c 0.99 -g 1 -p 0

* Mean Squared Error (MSE_TEST) = 0.0185

* Squared correlation coefficient = 0.6394

SVR RBF Kernel Result for CP2:

Param: -s 3 -t 2 -c 0.99 -g 0.25 -p 0

* Mean Squared Error (MSE_TEST) = 0.0037

* Squared correlation coefficient = 0.9423

SVR Sigmoid Kernel Result for CP2:

Param: -s 3 -t 3 -c 0.99 -g 0.5 -p 0

* Mean Squared Error (MSE_TEST) = 0.0040

* Squared correlation coefficient = 0.9350

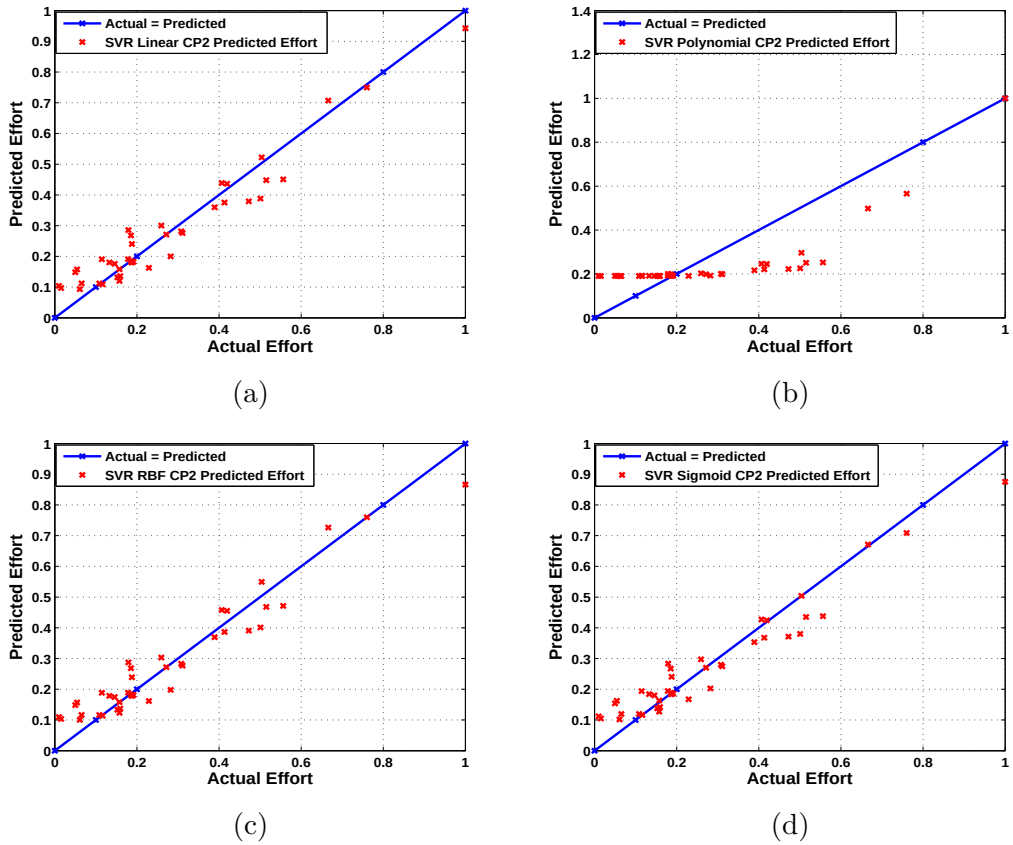


Figure 3.7: SVR Linear, Polynomial, RBF and Sigmoid Kernel based Effort Estimation Model for CP2 using 40 Project Dataset

The proposed model generated values for CP2 using the SVR linear, polynomial,

RBF and sigmoid kernel have been as shown in Figures 3.7a, 3.7b, 3.7c and 3.7d respectively. These figures display the variation of actual effort and the predicted effort obtained for CP2 using the various SVR kernel methods technique taking into consideration of 40 project dataset. In these graphs, it is clearly shown that the data points are very little dispersed than the regression line. Hence the correlation is higher. While comparing the dispersion of data points from the predicted model in the above graphs, it is clearly visible that in case of CP1 and CP2, the data points are less dispersed for SVR RBF kernel based model than other models. Hence, this model exhibits less error values and higher prediction accuracy value.

The ninety models generated to perform model selection operation using 30 project dataset for CP1 and CP2 are provided in the following tables.

Table 3.16: Validation Errors Obtained Using SVR Linear Kernel for CP1

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	**0.0037**	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-6}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-5}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-4}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-3}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-2}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-1}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662
2^{0}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662
2^{1}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662
2^{2}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662
2^{3}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662
2^{4}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662
2^{5}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662
2^{6}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662
2^{7}	0.0037	0.1662	0.1662	0.1662	0.1662	0.1662

Table 3.17: Validation Errors Obtained Using SVR Polynomial Kernel for CP1

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.2052	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-6}	0.2052	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-5}	0.2052	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-4}	0.2051	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-3}	0.2046	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-2}	0.2009	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-1}	0.1720	0.1662	0.1662	0.1662	0.1662	0.1662
2^{0}	0.0478	0.1662	0.1662	0.1662	0.1662	0.1662
2^{1}	0.0344	0.1662	0.1662	0.1662	0.1662	0.1662
2^{2}	0.0347	0.1662	0.1662	0.1662	0.1662	0.1662
2^{3}	0.0347	0.1662	0.1662	0.1662	0.1662	0.1662
2^{4}	0.0347	0.1662	0.1662	0.1662	0.1662	0.1662
2^{5}	0.0347	0.1662	0.1662	0.1662	0.1662	0.1662
2^{6}	0.0345	0.1662	0.1662	0.1662	0.1662	0.1662
2^{7}	**0.0335**	0.1662	0.1662	0.1662	0.1662	0.1662

Tables 3.16 and 3.17 show the validation error of ninety numbers of models generated for CP1 using SVR linear kernel and SVR polynomial kernel respectively based on the value of ϵ and γ for 30 project dataset. For SVR Linear kernel, *0.0037* value has been chosen as the minimum validation error. Hence based on the minimum validation error, the best model is C = 0.86782, $\gamma = 0.0078125$ and $\epsilon = 0$. For SVR Polynomial kernel, *0.0335* value has been chosen as the minimum validation error. Hence based on the minimum validation error, the best model is C = 0.86782, $\gamma = 128$ and $\epsilon = 0$.

Tables 3.18 and 3.19 show the validation error of ninety numbers of models generated for CP1 using SVR RBF kernel and SVR Sigmoid kernel respectively based on the value of ϵ and γ for 30 project dataset. For SVR RBF kernel, *0.0011* value has been chosen as the minimum validation error. Hence based on the minimum validation error, the best model is C = 0.86782, $\gamma = 8$ and $\epsilon = 0$. For SVR Sigmoid kernel, *0.0029* value has been chosen as the minimum validation error. Hence based on the minimum validation error, the best model is C = 0.86782, $\gamma = 1$ and $\epsilon = 0$.

Table 3.18: Validation Errors Obtained Using SVR RBF Kernel for CP1

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.1880	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-6}	0.1717	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-5}	0.1416	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-4}	0.0907	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-3}	0.0248	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-2}	0.0033	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-1}	0.0019	0.1662	0.1662	0.1662	0.1662	0.1662
2^{0}	0.0013	0.1662	0.1662	0.1662	0.1662	0.1662
2^{1}	0.0012	0.1662	0.1662	0.1662	0.1662	0.1662
2^{2}	0.0011	0.1662	0.1662	0.1662	0.1662	0.1662
2^{3}	**0.0011**	0.1662	0.1662	0.1662	0.1662	0.1662
2^{4}	0.0017	0.1662	0.1662	0.1662	0.1662	0.1662
2^{5}	0.0034	0.1662	0.1662	0.1662	0.1662	0.1662
2^{6}	0.0075	0.1662	0.1662	0.1662	0.1662	0.1662
2^{7}	0.0095	0.1662	0.1662	0.1662	0.1662	0.1662

Table 3.19: Validation Errors Obtained Using SVR Sigmoid Kernel for CP1

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.1965	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-6}	0.1880	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-5}	0.1716	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-4}	0.1411	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-3}	0.0894	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-2}	0.0227	0.1662	0.1662	0.1662	0.1662	0.1662
2^{-1}	0.0039	0.1662	0.1662	0.1662	0.1662	0.1662
2^{0}	**0.0029**	0.1662	0.1662	0.1662	0.1662	0.1662
2^{1}	0.0029	0.1662	0.1662	0.1662	0.1662	0.1662
2^{2}	0.0143	0.1662	0.1662	0.1662	0.1662	0.1662
2^{3}	0.0737	0.1662	0.1662	0.1662	0.1662	0.1662
2^{4}	0.5490	0.1662	0.1662	0.1662	0.1662	0.1662
2^{5}	4.0330	0.1662	0.1662	0.1662	0.1662	0.1662
2^{6}	7.5141	0.1662	0.1662	0.1662	0.1662	0.1662
2^{7}	8.7455	0.1662	0.1662	0.1662	0.1662	0.1662

Based on model parameters value, the model has been again trained and tested using training and testing data set respectively to estimate the effort.

Table 3.20: Validation Errors Obtained Using SVR Linear Kernel for CP2

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	**0.0033**	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-6}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-5}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-4}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-3}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-2}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-1}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734
2^{0}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734
2^{1}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734
2^{2}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734
2^{3}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734
2^{4}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734
2^{5}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734
2^{6}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734
2^{7}	0.0033	0.1734	0.1734	0.1734	0.1734	0.1734

Table 3.21: Validation Errors Obtained Using SVR Polynomial Kernel for CP2

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.2014	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-6}	0.2014	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-5}	0.2014	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-4}	0.2013	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-3}	0.2009	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-2}	0.1971	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-1}	0.1682	0.1734	0.1734	0.1734	0.1734	0.1734
2^{0}	0.0389	0.1734	0.1734	0.1734	0.1734	0.1734
2^{1}	0.0322	0.1734	0.1734	0.1734	0.1734	0.1734
2^{2}	0.0325	0.1734	0.1734	0.1734	0.1734	0.1734
2^{3}	0.0325	0.1734	0.1734	0.1734	0.1734	0.1734
2^{4}	0.0325	0.1734	0.1734	0.1734	0.1734	0.1734
2^{5}	0.0325	0.1734	0.1734	0.1734	0.1734	0.1734
2^{6}	0.0323	0.1734	0.1734	0.1734	0.1734	0.1734
2^{7}	**0.0314**	0.1734	0.1734	0.1734	0.1734	0.1734

Tables 3.20 and 3.21 show the validation error of ninety numbers of models generated for CP2 using SVR linear kernel and SVR polynomial kernel respectively based on the values of ϵ and γ for 40 project dataset. For SVR Linear kernel, *0.0033* value has been chosen as the minimum validation error. Based on the minimum validation error, the best model is C = 0.86782, $\gamma = 0.0078125$ and $\epsilon = 0$. For SVR Polynomial kernel, *0.0314* value has been chosen as the minimum validation error. Based on the minimum validation error, the best model is C = 0.86782, $\gamma = 128$ and $\epsilon = 0$.

Tables 3.22 and 3.23 show the validation error of ninety numbers of models generated for CP2 using SVR RBF kernel and SVR Sigmoid kernel respectively based on the values of ϵ and γ for 40 project dataset. For SVR RBF kernel, *0.0019* value has been chosen as the minimum validation error. Based on the minimum validation

Table 3.22: Validation Errors Obtained Using SVR RBF Kernel for CP2

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.1845	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-6}	0.1684	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-5}	0.1386	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-4}	0.0886	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-3}	0.0249	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-2}	0.0064	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-1}	0.0032	0.1734	0.1734	0.1734	0.1734	0.1734
2^{0}	0.0031	0.1734	0.1734	0.1734	0.1734	0.1734
2^{1}	0.0035	0.1734	0.1734	0.1734	0.1734	0.1734
2^{2}	0.0029	0.1734	0.1734	0.1734	0.1734	0.1734
2^{3}	**0.0019**	0.1734	0.1734	0.1734	0.1734	0.1734
2^{4}	0.0025	0.1734	0.1734	0.1734	0.1734	0.1734
2^{5}	0.0038	0.1734	0.1734	0.1734	0.1734	0.1734
2^{6}	0.0081	0.1734	0.1734	0.1734	0.1734	0.1734
2^{7}	0.0114	0.1734	0.1734	0.1734	0.1734	0.1734

Table 3.23: Validation Errors Obtained Using SVR Sigmoid Kernel for CP2

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.1928	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-6}	0.1845	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-5}	0.1683	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-4}	0.1382	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-3}	0.0873	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-2}	0.0229	0.1734	0.1734	0.1734	0.1734	0.1734
2^{-1}	0.0064	0.1734	0.1734	0.1734	0.1734	0.1734
2^{0}	**0.0036**	0.1734	0.1734	0.1734	0.1734	0.1734
2^{1}	0.0072	0.1734	0.1734	0.1734	0.1734	0.1734
2^{2}	0.0223	0.1734	0.1734	0.1734	0.1734	0.1734
2^{3}	0.0859	0.1734	0.1734	0.1734	0.1734	0.1734
2^{4}	0.5654	0.1734	0.1734	0.1734	0.1734	0.1734
2^{5}	4.0025	0.1734	0.1734	0.1734	0.1734	0.1734
2^{6}	7.4660	0.1734	0.1734	0.1734	0.1734	0.1734
2^{7}	8.6979	0.1734	0.1734	0.1734	0.1734	0.1734

error, the best model is C = 0.86782, $\gamma = 128$ and $\epsilon = 0$. For SVR Sigmoid kernel, *0.0036* value has been chosen as the minimum validation error. Based on the minimum validation error, the best model is C = 0.86782, $\gamma = 1$ and $\epsilon = 0$. Based on model parameters value, the model has been again trained and tested using training and testing data set respectively to estimate the effort.

After implementing the support vector regression based model using four different kernel methods for software effort estimation using 30 project dataset, the following results have been generated.

SVR Linear Kernel Result for CP1:

Param: -s 3 -t 0 -c 0.86782 -g 0.0078125 -p 0

* Mean Squared Error (MSE_TEST) = 0.0020

* Squared correlation coefficient = 0.9843

SVR Polynomial Kernel Result for CP1:

Param: -s 3 -t 1 -c 0.86782 -g 128 -p 0

* Mean Squared Error (MSE_TEST) = 0.0063

* Squared correlation coefficient = 0.9422

SVR RBF Kernel Result for CP1:

Param: -s 3 -t 2 -c 0.86782 -g 8 -p 0

* Mean Squared Error (MSE_TEST) = 0.0044

* Squared correlation coefficient = 0.9675

SVR Sigmoid Kernel Result for CP1:

Param: -s 3 -t 3 -c 0.86782 -g 1 -p 0

* Mean Squared Error (MSE_TEST) = 0.0030

* Squared correlation coefficient = 0.9771

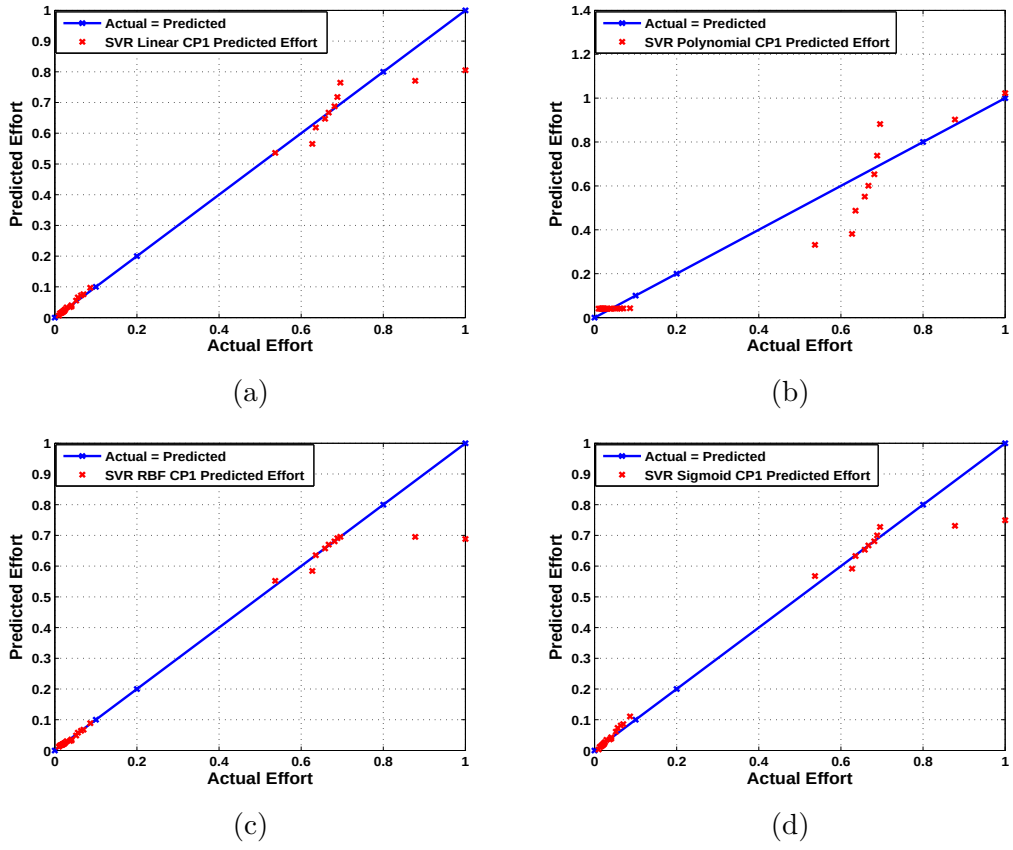


Figure 3.8: SVR Linear, Polynomial, RBF and Sigmoid Kernel based Effort Estimation Model for CP1 using 30 Project Dataset

The proposed model generated values for CP1 using the SVR linear, polynomial, RBF and sigmoid kernel have been plotted as shown in Figures 3.8a, 3.8b, 3.8c and 3.8d respectively. These figures display the variation of actual effort and the predicted effort obtained for CP1 using the four SVR kernel methods taking into consideration 30 project dataset.

SVR Linear Kernel Result for CP2:

Param: -s 3 -t 0 -c 0.86782 -g 0.0078125 -p 0

* Mean Squared Error (MSE_TEST) = 0.0026

* Squared correlation coefficient = 0.9797

SVR Polynomial Kernel Result for CP2:

Param: -s 3 -t 1 -c 0.86782 -g 128 -p 0

* Mean Squared Error (MSE_TEST) = 0.1225

* Squared correlation coefficient = 0.8781

SVR RBF Kernel Result for CP2:

Param: -s 3 -t 2 -c 0.86782 -g 8 -p 0

* Mean Squared Error (MSE_TEST) = 0.0041

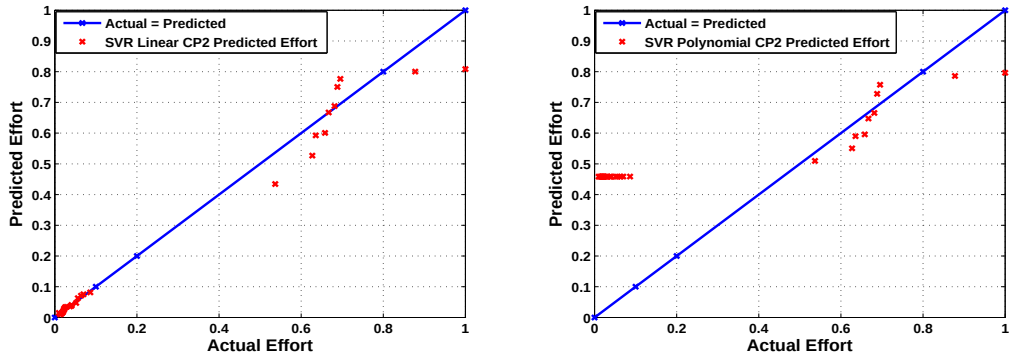
* Squared correlation coefficient = 0.9707

SVR Sigmoid Kernel Result for CP2:

Param: -s 3 -t 3 -c 0.86782 -g 1 -p 0

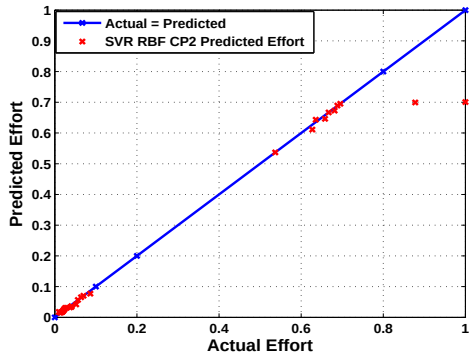
* Mean Squared Error (MSE_TEST) = 0.0027

* Squared correlation coefficient = 0.9800

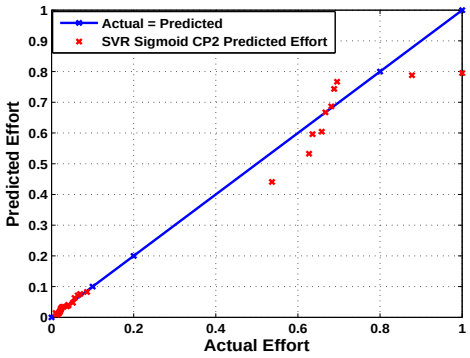


(a)

(b)



(c)



(d)

Figure 3.9: SVR Linear, Polynomial, RBF and Sigmoid Kernel based Effort Estimation Model for CP2 using 30 Project Dataset

The proposed model generated values for CP2 using the SVR linear, polynomial, RBF and sigmoid kernel have been as shown in Figures 3.9a, 3.9b, 3.9c and 3.9d respectively. These figures display the variation of actual effort and the predicted effort obtained for CP2 using the various SVR kernel methods technique taking into consideration 30 project dataset. In these graphs, it may be observed that the data points are very little dispersed than the regression line. Hence the correlation is higher. While comparing the dispersion of data points from the predicted model in the above graphs, it is observed that in case of CP1 and CP2, the data points are less dispersed

for SVR RBF kernel based model than other models. Hence, this model exhibits less error values and higher prediction accuracy value.

The *squared correlation coefficient*(r^2) is known as the *coefficient of determination*. It is one of the better means for evaluating the strength of correlation. It is the proportion of variance in actual effort that can be accounted for by knowing class point value for training data set. In the output generated, it is quite clearly mentioned that the *squared correlation coefficient* value for both RBF kernel and sigmoid kernel is very high (greater than 0.9). Hence it can be concluded that there is a strong positive correlation exists between the class point (CP1 & CP2) and the predicted effort required to develop the software i.e., a minor change in the class point value results in significant change in the predicted effort value.

3.5 Comparison

On the basis of results obtained, the estimated effort using SGB and various SVR kernel methods are compared. The result shows that in case of CP1 and CP2, effort estimation using SVR RBF kernel based model gives less error values and higher prediction accuracy value than those obtained using other machine learning models for both 40 and 30 project dataset. Hence, it is observed that SVR RBF Kernel technique performs better than other techniques.

Table 3.24: Comparison of Prediction Accuracy Values of Related Works

Related Articles	**Technique Used**	**PRED(25)**	
		CP1	**CP2**
Costagliola et al. [53]	**Regression Analysis**	**75%**	**83%**
Zhou and Liu [54]	**Regression Analysis**	**75%**	**83%**
Kanmani et al. [55]	**Neural Network**	**83%**	**87%**

Table 3.24 provides a comparative study of the results obtained by authors of different articles mentioned in the related work section. The performance of techniques used by those authors have been compared by measuring their prediction accuracy (PRED) values. The result shows that, the authors of first two articles provide same prediction accuracy values for CP1 and CP2; where as the author of third article shows some improvement in the prediction accuracy value. Finally, the results obtained in related work section is compared with results of proposed approaches, which is shown in Tables 3.25, 3.26, 3.27 and 3.28.

Tables 3.25 and 3.26 display the final comparison of RMSE, MAE, MMRE, MMER

Table 3.25: Comparison of Results of SGB & Various SVR Kernels for CP1 using 40 Dataset

	RMSE	**MAE**	**MMRE**	**MMER**	**PRED(25)**
SGB	0.0704	0.0573	0.7863	0.2557	87.5000
SVR Linear	0.0692	0.0555	0.7708	0.2542	90
SVR Polynomial	0.1531	0.1164	1.2500	0.5206	85
SVR RBF	0.0661	0.0523	0.7082	0.2541	90
SVR Sigmoid	0.0704	0.0573	0.7863	0.2557	87.5000

Table 3.26: Comparison of Results of SGB & Various SVR Kernels for CP2 using 40 Dataset

	RMSE	**MAE**	**MMRE**	**MMER**	**PRED(25)**
SGB	0.0630	0.0491	0.6767	0.2287	90
SVR Linear	0.0600	0.0483	0.6611	0.2264	90
SVR Polynomial	0.1359	0.1066	1.2325	0.4722	85
SVR RBF	0.0576	0.0463	0.6323	0.2259	92.5000
SVR Sigmoid	0.0630	0.0491	0.6768	0.2287	90

Table 3.27: Comparison of Results of SGB & Various SVR Kernels for CP1 using 30 Dataset

	RMSE	**MAE**	**MMRE**	**MMER**	**PRED(25)**
SGB	0.0546	0.0211	0.1260	0.1787	83.3333
SVR Linear	0.0446	0.0206	0.0737	0.0796	93.3333
SVR Polynomial	0.0794	0.0488	0.6171	0.3819	36.6666
SVR RBF	0.0445	0.0201	0.0730	0.0786	96.6666
SVR Sigmoid	0.0546	0.0211	0.1261	0.1800	83.3333

Table 3.28: Comparison of Results of SGB & Various SVR Kernels for CP2 using 30 Dataset

	RMSE	**MAE**	**MMRE**	**MMER**	**PRED(25)**
SGB	0.0521	0.0264	0.1371	0.1441	86.6666
SVR Linear	0.0514	0.0267	0.1336	0.1346	86.6666
SVR Polynomial	0.3500	0.3042	11.8321	0.6482	33.3333
SVR RBF	0.0638	0.0195	0.0970	0.0925	93.3333
SVR Sigmoid	0.0520	0.0264	0.1369	0.1440	86.6666

and PRED(25) values for SGB and different SVR kernel methods for 40 project dataset. Tables 3.27 and 3.28 display the final comparison of RMSE, MAE, MMRE, MMER and PRED(25) values for SGB and different SVR kernel methods for 30 project dataset. While comparing the obtained results with the results provided in related work section i.e., in Table 3.24, it can be observed that the obtained results from proposed models provide better prediction accuracy values than the results obtained from models given in related work section. The results obtained from different proposed models show that in case of CP1 and CP2, effort estimation using SVR RBF kernel gives less values of RMSE, MAE, MMRE, MMER and higher value of prediction accuracy than those obtained using other machine learning models for both 40 and 30 project dataset.

Figures 3.10a and 3.10b display the box plot for CP1 and CP2 using 40 project dataset respectively. Figures 3.10c and 3.10d display the box plot for CP1 and CP2 using 30 project dataset respectively. These figures are used to illustrate the spread and differences of samples, with the help of their corresponding error values generated using SGB, and different SVR kernel methods.

Table 3.29: Comparison of Effect Size Test of Proposed Models for CP1 using 40 Project Dataset

	Effect Size Test	
	Cohen's d	**Glass's Δ**
SGB vs. SVR-Linear	0.0883	0.0883
SGB vs. SVR-Polynomial	0.2448	0.2966
SGB vs. SVR-RBF	0.0629	0.0614
SGB vs. SVR-Sigmoid	0.0887	0.0901

Table 3.30: Comparison of Statistical Significance and Effect Size Test of Proposed Models for CP2 using 40 Project Dataset

	Effect Size Test	
	Cohen's d	**Glass's Δ**
SGB vs. SVR-Linear	0.0366	0.0361
SGB vs. SVR-Polynomial	0.1974	0.2281
SGB vs. SVR-RBF	0.0546	0.0542
SGB vs. SVR-Sigmoid	0.0194	0.0199

In order to affirm the robustness of the proposed models, the effect size [131] tests such as Cohen's d test and Glass's Δ test between diverse proposed models are processed considering absolute residuals as demonstrated in Tables 3.29, 3.30, 3.31

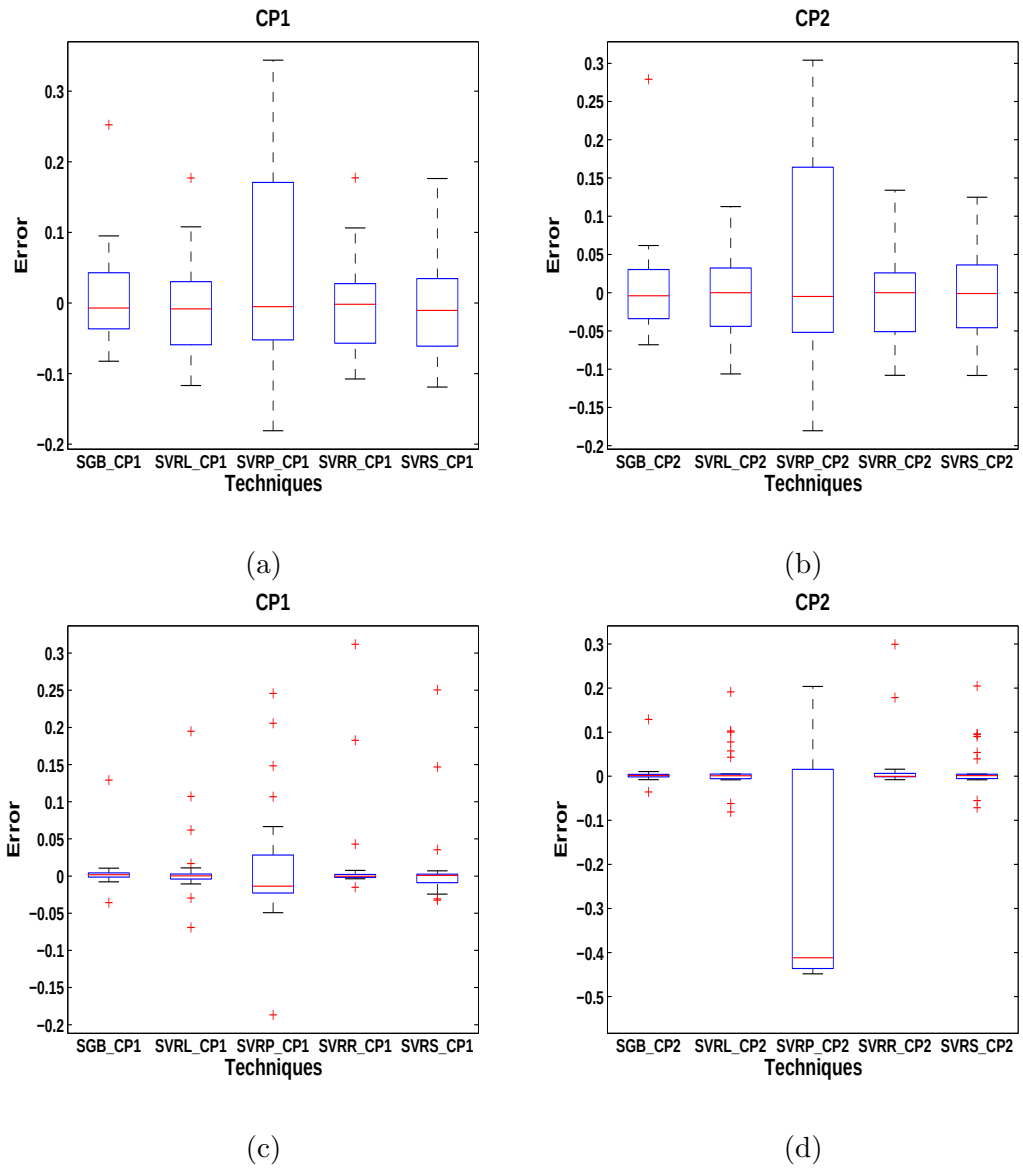


Figure 3.10: Boxplot of Error Values for 40 and 30 Project Datasets

Table 3.31: Comparison of Statistical Significance and Effect Size Test of Proposed Models for CP1 using 30 Project Dataset

	Effect Size Test	
	Cohen's d	**Glass's Δ**
SGB vs. SVR-Linear	0.0123	0.0124
SGB vs. SVR-Polynomial	0.0249	0.0249
SGB vs. SVR-RBF	0.0412	0.0426
SGB vs. SVR-Sigmoid	0.0166	0.0170

Table 3.32: Comparison of Statistical Significance and Effect Size Test of Proposed Models for CP2 using 30 Project Dataset

	Effect Size Test	
	Cohen's d	**Glass's Δ**
SGB vs. SVR-Linear	0.0279	0.0283
SGB vs. SVR-Polynomial	0.1360	0.4125
SGB vs. SVR-RBF	0.0400	0.0414
SGB vs. SVR-Sigmoid	0.0308	0.0314

and 3.32 for CP1 and CP2 using 40 and 30 project respectively. From the results provided in Tables 3.29, 3.30, 3.31 and 3.32, it is evident that the effect size is mostly small in all the cases, for class point dataset.

3.6 Summary

In the literature, it is observed that a good number of approaches have been considered by researchers and practitioners to calculate the effort required to develop a given software product. In this chapter, an attempt has been made to estimate the effort required for developing object-oriented software by using class point approach. The class point model is enhanced using the SGB and four SVR kernel techniques and the results obtained are compared with existing as well as proposed models. The results show that the SVR RBF Kernel-based effort estimation model possesses lower RMSE, MAE, MMRE, MMER and higher prediction accuracy. The computations for the procedure were implemented and the outputs were generated using MATLAB software.

Chapter 4

Use Case Point Approach for Software Effort Estimation using Machine Learning Techniques

4.1 Introduction

Due to the increasing complexity of software development activities, the need for effective effort estimation techniques has arisen. Underestimation leads to disruption in the projects estimated cost and delivery. On the other hand, overestimation causes outbidding and financial losses in business [124]. The job of software effort estimation is a critical one in the early stages of the software development life cycle, when the details of requirements are usually not clearly identified. Hence, effort estimation during early stage of software development life cycle (SDLC) plays a vital role for determining whether a project is feasible in terms of a cost-benefit analysis [132, 133] or not.

Use Case Point (UCP) approach relies on the use case diagram of UML paradigm for effort estimation of a given software product [7]. UCP helps in providing a comprehensive effort estimation from the design phase itself. The total number of UCP is measured by ascertaining the total no. of use cases as well as actors and then multiplying each of them with their corresponding complexity factors. Each use case and actor are classified into one of the three classes such as simple, average and complex. The number of transactions per use case helps in determining its complexity value. The UCP model has broadly been utilized with in the most recent decade [134]; still it possesses certain limitations. It assumes that the software size and effort are directly proportional to each other. Due to this reason, the software effort equation provided by UCP is not well accepted by software industries. Various optimization techniques help in improving the accuracy of effort estimation. In this study, random forest and support vector regression techniques are employed to

handle the confinements of the UCP model and to enhance prediction accuracy of software effort estimation. The results obtained applying these techniques-based effort estimation model is then measured against the results achieved applying Multi-Layer Perceptron (MLP), Radial Basis Function Network (RBFN), Stochastic Gradient Boosting (SGB) and Log-Linear Regression (LLR) models so as to critically evaluate their performances.

4.2 Methodologies Used

The methodologies that are utilized within this study to compute the effort needed to create a software product are described below:

4.2.1 Use Case Point (UCP) Approach

The Use Case Point (UCP) approach was initially proposed by Gustav Karner in 1993 [135]. This approach is an extension of Function Point Approach (FPA) and Mk II FPA. If the idea about the problem domain, system size and architecture is known, then an early effort estimation focused around use cases could be made. Figure 4.1 demonstrates the different steps taken into consideration to compute the total no. of use case points [136, 137].

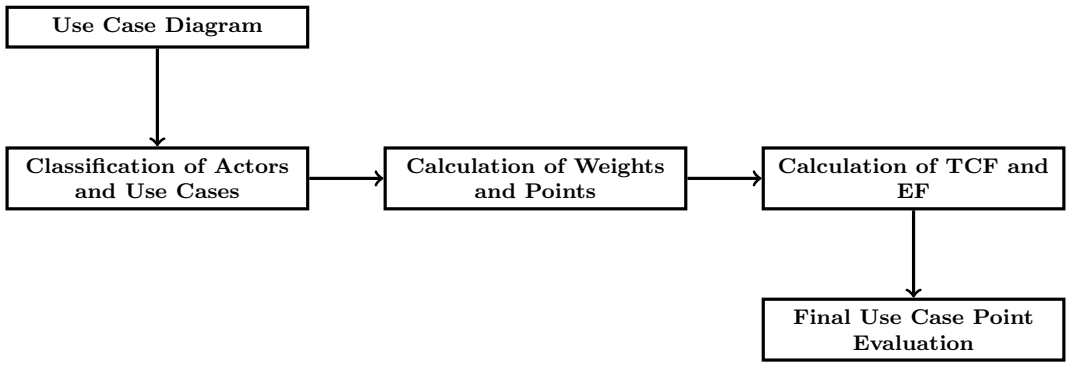


Figure 4.1: Steps to Calculate Use Case Points

The use case point approach can be implemented using the following steps:

Classification of Actors and Use Cases

This step deals with classifying the actors in a use case diagram as simple, average or complex. An actor that represents a system with a well defined Application Programming Interface (API), is considered as simple. An actor that communicates with the system through a protocol, is classified as average. An actor is classified as complex, if it can represent a person who is interacting with system through a Web

page or Graphical User Interface (GUI). Each actor type is assigned with a weighting factor as shown in Table 4.1.

Table 4.1: Assignment of Weighting Factors to Each Actor [135]

Type of Actor	**Corresponding Weight**
Simple	1
Average	2
Complex	3

Similarly, a use case is classified into either simple or average or complex type. Classification of the use case depends on the number of transactions characterized in the description of use case along with secondary scenarios. A use case is considered *Simple*, if it uses less than four number of transactions to interact and also uses only a single database object. A use case is classified as *Average*, if it involves four to seven number of transactions and uses two or more database objects. A use case that involves more than seven number of transactions for processing and requires greater than or equal to three database objects, is considered as *Complex*. The complexity of a use case is characterized and weighted using the value given in Table 4.2.

Table 4.2: Assignment of Weighting Factors to Each Use Case [135]

Type of Use Case	**No. of Transactions**	**Corresponding Weight**
Simple	<= 3	5
Average	4 to 7	10
Complex	>= 7	15

Calculation of Weights and Points

The total Unadjusted Actor Weights (UAW) is computed by calculating the number of actors of each type (by degree of complexity), multiplying each total with corresponding weighting factor, and finally summed up the products. The UAW is determined as follows:

$$UAW = \sum_{i=1}^{3} N_i \times W_i \tag{4.1}$$

where N_i is the number of actors of variety i and W_i is their corresponding complexity weight. Similarly each type of use case is then multiplied by their corresponding weighting factor, and the products are summed up to get the total Unadjusted Use Case Weights (UUCW). The UUCW is determined as follows:

$$UUCW = \sum_{j=1}^{3} P_j \times X_j \tag{4.2}$$

where P_j is the number of use cases of variety j and X_j is their corresponding complexity weight. Finally, the Unadjusted Use Case Points (UUCP) is obtained by adding UAW value with the UUCW value.

$$UUCP = UAW + UUCW \tag{4.3}$$

Calculation of Technical Complexity Factor (TCF) and Environmental Factor (EF)

The UUCP value obtained from the above equation is altered based on the weights alloted to thirteen technical factors and eight environmental factors [70, 138] as indicated in Tables 4.3 and 4.4. Each factor is rated between the range 0 to 5 depending on its expected impact over the project. A rating of 0 signifies that the corresponding factor is unimportant for the project. Similarly, a rating of 5 signifies it is essential. Technical factors (TFactors) contribute to the complexity of the project; while Environmental factors contribute to the team efficiency and productivity.

Table 4.3: Technical Factors [135]

Factor ID (T_i)	**Complexity Factors**	**Corresponding Weight**
T_1	Concurrency	1
T_2	Code Reusability	1
T_3	Special Training facilities	1
T_4	Distributed System	2
T_5	End-user Efficiency	1
T_6	Installation Ease	0.5
T_7	Portability	2
T_8	Complex Internal Processing	1
T_9	Changeability	1
T_{10}	Operational Ease, Usability	0.5
T_{11}	Provide Direct Access to Third Parties	1
T_{12}	Special Security Features	1
T_{13}	Application Performance Objectives in either Response or Throughput	1

Table 4.4: Environment Factors [135]

Factor ID(E_i)	**Efficiency and Productivity Factors**	**Corresponding Weight**
E_1	Requirements Stability	2
E_2	Motivation	1
E_3	Experience in Handling Applications	0.5
E_4	Part-time Workers	-1
E_5	Capability of Analysts	0.5
E_6	Programming Language Difficulty	-1
E_7	Familiarity with Rational Unified Process	1.5
E_8	Experience in Developing Object-oriented Software	1

These factors are multiplied by the UUCP to calculate the final UCP. The TCF

is obtained by multiplying the technical factors (T1- T13) by their corresponding weight and after that, summing all the obtained values to calculate *TFactor*. *TFactor* is calculated as follows:

$$TFactor = \sum_{i=1}^{13} T_i \times W_i \tag{4.4}$$

where T_i is a factor that takes values between 0 and 5 and W_i is their corresponding complexity weight. The accompanying equation gives TCF:

$$TCF = 0.6 + (0.01 * TFactor) \tag{4.5}$$

Similarly, EF is calculated by multiplying the environmental factors (F1-F8) by their corresponding weight and after that summing all the obtained values to calculate *EFactor*. *EFactor* is calculated as follows:

$$EFactor = \sum_{i=1}^{8} E_i \times W_i \tag{4.6}$$

where E_i is a factor that takes values between 0 and 5 and W_i is their corresponding complexity weight. The accompanying equation gives EF:

$$EF = 1.4 + (-0.03 * EFactor) \tag{4.7}$$

Use Case Point (UCP) Calculation

The final adjusted UCP are calculated as follows:

$$UCP = UUCP * TCF * EF \tag{4.8}$$

The final use case point is then considered as an input argument to random forest and four SVR kernel models to calculate effort.

4.3 Proposed Approach

Datasets such as Albrecht, COCOMO, Desharnais, and NASA, shared publicly are not applicable for the proposed work, because the size metric used in these datasets is either Source Lines of Code (SLOC) or function points. Due to this reason, a set of questionnaires were prepared by Nassif et al. [24] and it could help in obtaining industrial data without the help of UML diagrams. Using this process, a one hundred forty nine projects data based on use case point were obtained as considered by Nassif et al. [24], and is used in the proposed approach. These dataset is collected from three different sources, i.e., fifty from ISBSG [10], sixty five from Western University and

thirty four from a medium-sized company. In the dataset table, every row contains four columns. The first column indicates software size which is calculated as total number of use case points required to complete the project. Hence, in this case the unit of software size is use case point. The second column indicates team productivity rate, the third column indicates project complexity. The fourth column represents the actual effort required to complete that project, which is calculated in person-hours. The statistical profile of dataset based on Use Case Point Approach is depicted in Table 4.5. Figure 4.2 depicts the relationship between software size (total number

Table 4.5: Statistical Profile of Datasets based on Use Point Approach

Project Type	**Minimum**	**Maximum**	**Mean**	**Median**	**Std. Dev.**	**Skewness**	**Kurtosis**
149 Project Dataset	122	224890	14140.79	2530	33824.38	4.42	21.04

of use case point) and software effort (person-hours) based on UCP approach using 149 project dataset. From these figures, it is observed that the 149 project dataset

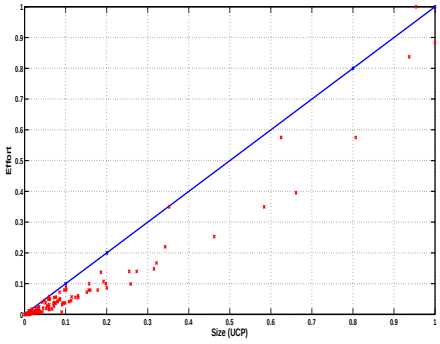
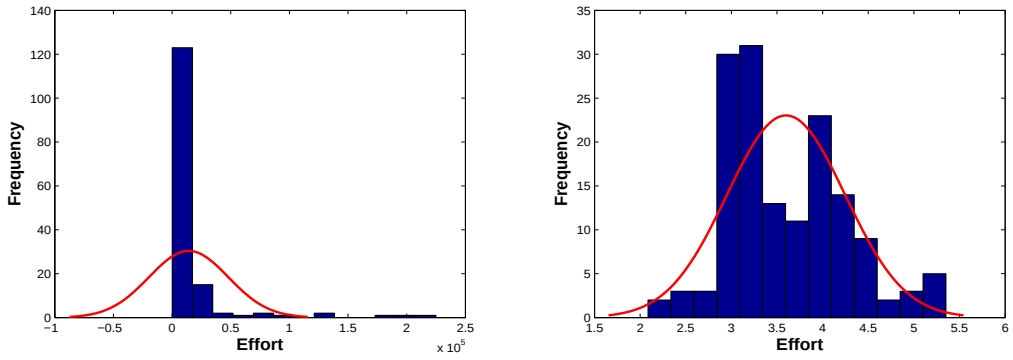


Figure 4.2: Software Size vs. Effort Graph based on UCP approach using 149 project dataset

based on UCP approach contains few number of outliers. From Table 4.5, it has been observed that the dataset is not normally distributed based on the values of the skewness and kurtosis. Hence, in order to make the data normally distributed, logarithmic transformation is applied over the dataset.

Figures 4.3a and 4.3b display the histogram of effort value before and after applying logarithmic transformation respectively. From these figures, it can be observed that the data are now more normally distributed after applying logarithmic transformation. A sample of ten projects data out of a one hundred forty nine projects data is provided in Table 4.6 for reference.

Out of these, initially software size, productivity and complexity are considered as input parameters to the machine learning models in order to assess their influences over predicted effort value. Then, the input argument having the maximum influence



(a) (b)

Figure 4.3: Histogram of Effort value before and after Logarithmic Transformation

Table 4.6: Ten Sample Project Dataset

Project No.	Software Size	Team Productivity	Project Complexity	Actual Effort
1	**13.5**	**5**	**3**	**122**
2	**18**	**4**	**3**	**296**
3	**20**	**3**	**1**	**360**
4	**28**	**4**	**1**	**170**
5	**31**	**37**	**3**	**507**
6	**34**	**32**	**2**	**972**
7	**38**	**33**	**3**	**752**
8	**39**	**35**	**2**	**890**
9	**39**	**27**	**4**	**1024**
10	**39**	**35**	**2**	**1209.6**

is selected as the final input parameter to the model for calculating predicted effort. The utilization of such a dataset helps to calculate software development effort and provides introductory test information for the viability of the UCP. These data are utilized to obtain the random forest technique-based effort estimation model. Figure 4.4 presents the proposed steps considered, in order to evaluate the effort utilizing the random forest technique.

To compute the software development effort, the accompanying steps are followed:

Proposed Steps for Software Effort Estimation

1. **Collection of Software Size, Productivity, Complexity and Actual Effort Values**: The software size i.e., total number of use case points required to complete the project, productivity, complexity and actual effort values for one hundred forty nine projects are collected from the literature [24]. Collected size, productivity and complexity values are used as input arguments initially to

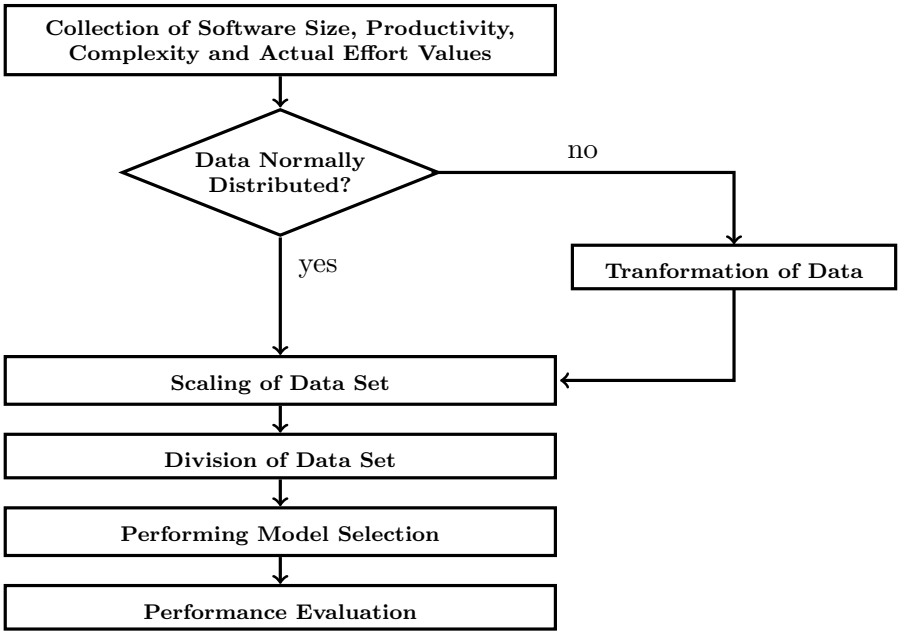


Figure 4.4: Proposed Steps for Software Effort Estimation Purpose Applying RF and SVR Kernel Techniques

RF model for calculating impact of each variable on the predicted effort value. But after calculating the impact of each variable, the highest impact variable is considered as final input argument to the RF model.

2. **Data Normally Distributed?**: The statistical analysis of the collected dataset has been performed. It is verified as to whether the collected dataset follows normal distribution or not, based on the values of skewness and kurtosis. If data are normally distributed, then it will directly proceed to the data normalization step. Otherwise, the data need to be transformed to make it more normally distributed.

3. **Transformation of Data**: If the dataset is not normally distributed, then the logarithmic transformation method has been applied over the dataset to make it normally distributed. Histograms have been plotted to properly verify the distribution of data before and after transformation.

4. **Scaling of Dataset** : The values taken as input arguments are individually scaled within the range 0 to 1. Let S represents the complete dataset and s represents a record in the S. Then the normalized value of a record 's' is obtained by considering the following equation:

$$Normalized(s) = \frac{s - \min(S)}{\max(S) - \min(S)} \tag{4.9}$$

where
$\min(S)$ = min value in S.
$\max(S)$ = max value in S.
if $\min(S)$ is same as $\max(S)$, then Normalized(s) value is assigned as 0.5.

5. **Division of dataset**: Total no. of data are divided into two subsets i.e., training set and test set for both RF and SVR Kernel techniques. Random forest have randomness in input data and in splitting at nodes. Hence, in case of RF technique, initially an arbitrary random vector is selected to provide randomness in input data and to start the implementation process. Then, the data are divided using this arbitrary random vector.

6. **Performing Model Selection**: In case of RF technique, prediction results vary according to random vector. So an evaluation function, (1- MMER + Prediction Accuracy) is used to find a random vector. The random vector, which provides optimum value for the evaluation function is considered as final random vector. Then, by using this final random vector, results are being predicted.

   Similarly, in case of SVR kernel-based effort estimation model, the model which provides the least value than the other generated models based on the minimum validation error criteria has been selected to perform other operations. The tunable parameters have been selected to find the best parameter C and γ using a five-fold cross validation procedure . Based on the minimum validation error, the best model has been selected and the corresponding value of γ and ϵ value is found out. The final model selected based on best parameter of C, ϵ and γ has been trained using all training samples. The output of this step is the trained SVM model providing predicted response values for test inputs.

7. **Performance Evaluation**: In this study, the Mean Magnitude of Error Relative to the estimate (MMER) and the Prediction Accuracy (PRED(x)) are the two measures used to evaluate the performance of the model for test samples. Results obtained from proposed model-based on RF and SVR Kernel techniques are then evaluated against existing results to access its performance accuracy.

4.3.1 Example

A sample dataset of 10 projects has already been provided in Table 4.6. These data are used in this section for demonstration purpose of proposed steps for software effort estimation using RF model. The first column represents software size, which is calculated in terms of number of use case point required to complete the project.

The second column represents the team productivity. EFactors provided in Table 4.4 contribute to the calculation of team efficiency and productivity. The third column represents project complexity. TFactors provided in Table 4.3 contribute to the calculation of complexity of the project. Finally the fourth column represents actual effort required to complete the project measured in person-hours.

After collecting the required project data, the second step in the proposed approach section deals with the normalization process of dataset. Hence after normalization, the following values will be generated.

Table 4.7: Normalized Project Dataset

Project No.	Normalized Software Size	Normalized Team Productivity	Normalized Project Complexity	Normalized Actual Effort
1	0.0001	0.1081	0.5000	0.0001
2	0.0015	0.0811	0.5000	0.0008
3	0.0021	0.0541	0.0001	0.0011
4	0.0048	0.0811	0.0001	0.0002
5	0.0058	0.9730	0.5000	0.0017
6	0.0068	0.8378	0.2500	0.0038
7	0.0081	0.8649	0.5000	0.0028
8	0.0084	0.9189	0.2500	0.0034
9	0.0084	0.7027	0.7500	0.0040
10	0.0084	0.9189	0.2500	0.0048

In this case, 'max' and 'min' values are forwarded out according to original one hundred forty nine projects dataset not this sample data-set. Hence. the normalized values shown in Table 4.7 are also obtained as per the original dataset. They are not based on the above ten sample dataset. The next step deals with selection of an arbitrary random vector. For example:
random vector = 6 3 7 8 5 1 2 4 9 10

Using this random vector, the dataset is divided into training and testing data. Then using the process given in experimental details section, the RF-based effort estimation model is applied to predict the effort value. These predicted effort values are compared with their corresponding actual normalized effort values to calculate the model accuracy. Finally, the performance evaluation process of different models is being carried out with help of various performance measures.

4.4 Experimental Details

In this study, for implementing the proposed methodology, dataset having a one hundred forty nine projects data from literature [24] are being used. An exhaustive depiction about the dataset has been given in the proposed approach section. After computing the no. of use case points, the dataset are then scaled. The scaled dataset

is split into two subsets i.e., *training set* and *test set*. The *training set* is utilized for learning purpose; whereas the *test set* is utilized just for evaluating the accuracy of prediction of the trained model. In this study, 80% of data are used for training and rest are used for testing. The reckonings were actualized, and the yields were produced using MATLAB with the help of an existing random forest library [139]. The number of trees is provided as the parameter to run the library. The default value is considered to be five hundred.

4.4.1 Model Design using Random Forest Technique

The Brieman's algorithm is popularly used to implement the random forest technique [19]. In order to obtain a random forest technique-based effort estimation model, the steps presented underneath are taken into consideration. These proposed steps help in constructing each tree, while using random forest technique.

Steps of Proposed Algorithm:

1. Let F be the number of trees in the forest. A Dataset of D points $(x_1, y_1)(x_2, y_2)....(x_D, y_D)$ is considered.
2. Each tree of the forest should be grown as follows: Steps from i to vii should be repeated f times to create F number of trees.
    i. Let N be the no. of training cases, and M be the no. of variables in the classifier.
    ii. To select training set for the tree, a random sample of n cases - yet with substitution, from the original data of all N accessible training cases is chosen. Whatever is left of the cases are utilized to evaluate the error of the tree, by foreseeing their classes.
    iii. A RF tree T_f is developed to the loaded data, by repeatedly rehashing the accompanying steps for every terminal node of the tree, till the minimum node size n_{min} is arrived. Keeping in mind the end goal to make more randomness, distinctive dataset for each one trees is made.
    iv. The no. of input variables m is selected to discover the choice at a tree node. The value of m ought to be substantially short of what M.
    v. For each tree node, m number of variables should be randomly chosen on which the decision at that node is based.
    vi. The best split focused around these m variables in the training set is calculated. The value of m ought to be held consistent throughout the development of the forest. Each tree should be fully grown and not pruned.

vii. Then, the results of ensemble of trees $T_1, T_2, ..., T_f, ..., T_F$ are collected.

3. The input vector should be put down for each of the trees in the forest. In regression, it is the average of the individual tree predictions.

$$Y^F(x) = 1/F \sum_{f=1}^{F} T_f(x) \tag{4.10}$$

where

$Y^F(x)$ is the predicted value for the input vector x.

$T_1(x), T_2(x), ..., T_f(x)$ represents prediction value of individual trees.

There are various data objects generated by random forest technique, which need to be considered while implementing random forest technique for software effort estimation purpose. The results obtained from these data objects need to be evaluated in order to assess the performance achieved using random forest technique.

Variable Importance

The variable importance defines the contribution of a variable in achieving accurate prediction. It is calculated by taking into consideration its interaction with other variables. The error rate for each tree T, is calculated using the Out-of-Bag(OOB) data. Then, the permutation result of the OOB values is calculated for each variable v and the error is again calculated again using each tree. If the number of variables for implementing RF technique is very large, forests can be run once with all the variables. Then, by using only the most important variable from the initial run, forests can be run again to calculate the final predicted effort.

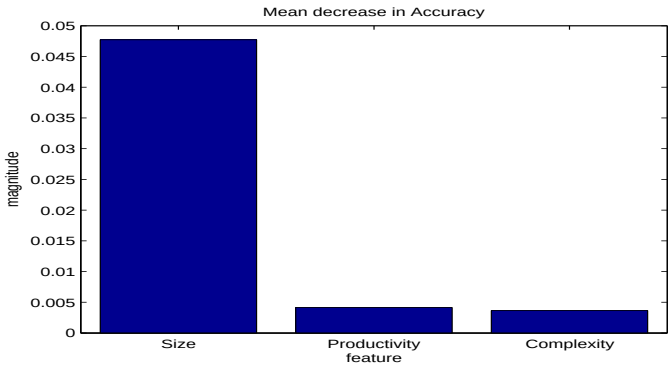


Figure 4.5: Variable Importance

Figure 4.5 displays the importance of three variables taken as input to the model for calculating the effort using random forest technique. The first column in the figure

represents the software size, the second column represents its productivity and the third column represents the complexity of the software. From the figure, it is clearly visible that the impact of the variable *software size* is highest for predicting the effort required to develop the software. Therefore, software size variable is finally chosen for predicting effort in the proposed approach.

The Out-Of-Bag (OOB) Error Estimate

The training set for a tree is produced by testing with substitution. During this process, something like one-third of the cases are left out of the sample. These cases are considered as out-of-bag (OOB) data. It helps in getting an impartial evaluation of the regression error value as the forest develops. OOB data also helps in getting estimation of variable importance. In RF, as the OOB is calculated internally during the run. Cross validation of data or a different test set to obtain an impartial evaluation of the test error is not required. The computation procedure for OOB is explained below.

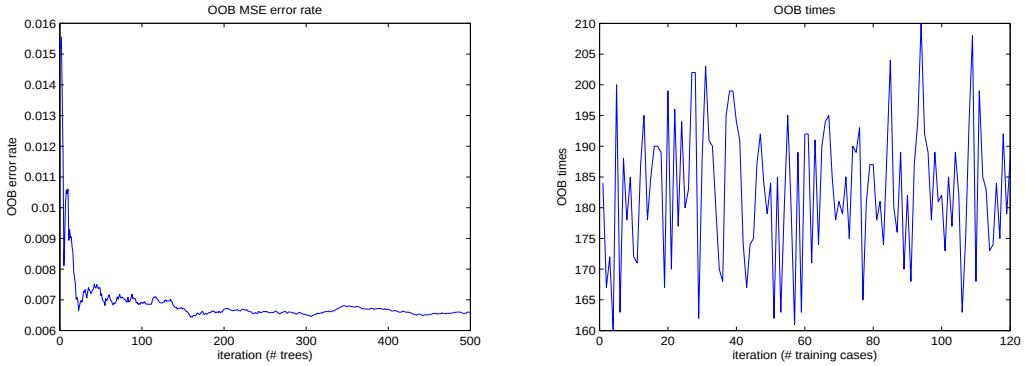
- During construction of each tree, an alternate bootstrap sample from the original data is used. Approximately, one-third of the cases from the bootstrap sample are left out and not used in the tree construction process. Hence, out of one hundred twenty data, eighty data are used in the tree construction process and rest forty data are used for testing the result.
- These OOB samples are put down the k^th tree to obtain a regression. Using this process, a test set is acquired for each one case.
- At the end, suppose 'j' be the predicted value that is acquired by computing the average prediction value of forest, each time case n was oob. The extent of times 'j' is not equivalent to the actual value of n averaged over all cases is called as the *out-of-bag error estimate*.

The RF prediction accuracy can be determined from these OOB data by using the following formula.

$$OOB - MSE = \frac{1}{F}\sum_{i=1}^{F}(y_i - \bar{y}_{iOOB})^2 \tag{4.11}$$

where $\bar{y}_{iOOB}$ represents the average prediction value of ith observation from all trees for which this observation has been OOB. F denotes the no. of trees in the forest and y_i represents the actual value.

Figure 4.6a displays the OOB error rate obtained for different number of trees used in the forest. From the figure, it is quite clearly visible that during initial phase (while the number of trees used are less), the OOB error rate obtained is maximum.



(a) (b)

Figure 4.6: OOB MSE Error Rate and Number of Times Out Of Bag Occurs

At the same time steadily with the increment of the amount of trees utilized within the forest, the OOB error rate converges to minimum value. After some period, OOB error rate remains constant.

Figure 4.6b displays the number of times, cases are out of bag for all training attributes. In this case, one hundred twenty number of training attributes have been considered.

Proximities

Proximity is one of the important data objects while calculating effort using RF technique. It measures the frequency of ending up the unique pairs of training samples in the same terminal node. It also helps in filling up the missing data in the dataset and calculating number of outliers.

Originally, a 'N × N' matrix is formed by the proximities. Once a tree is developed, all the data i.e., training data and out-of-bag data are put down the tree. Its proximities are increased by one, if it is found that two cases are in the same terminal node. Finally, the normalized values of the proximities are obtained by dividing with the number of trees.

Figure 4.7 describes the proximity value generated using random forest technique. A 120 × 120 matrix has been used for generating the above figure. From the figure, it is observed that, for diagonal elements, the proximity value is maximum (equals to one). But for all other elements, the proximity value is less than one. The symmetric portion adjacent to diagonal area represents other elements proximity values.

Complexity

In the proposed approach, 500 number of trees are taken into consideration for implementing RF technique. In the usual tree growing algorithm, all descriptors

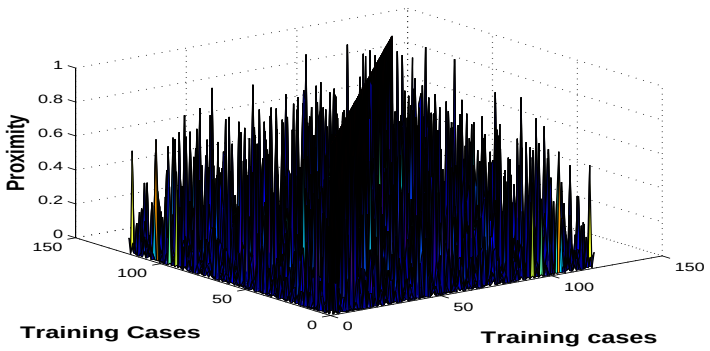


Figure 4.7: Proximity

are tested for their splitting performance at each node; while Random Forest only tests m try of the descriptors. Since m try is typically very small, the search is very fast.

To get the model complexity value for optimal prediction strength, pruning is usually done via cross validation for a single decision tree. This process can take up a significant portion of the computations. RF, on the other hand, does not perform any pruning at all. It is observed that in cases where there are an excessively large number of descriptors, RF can be trained in less time than a single decision tree. Hence, the RF algorithm may be observed as an efficient one.

Outlier

The cases that are expelled from the principal group of data and whose proximities to all different cases in the data are mostly small, are defined as *Outliers*. The concept of outliers can be revised by defining outliers relative to corresponding cases. In this way, an Outlier is the case whose proximities to all different cases are small. The average proximity is specified as:

$$\bar{P}(n) = \sum_{1}^{N} prox^2(n,k) \tag{4.12}$$

where 'n' and 'k' denote a training case in the regression and N represents the total no. of training cases in the forest. The raw outlier measure for case n is specified as:

$$nsample/\bar{P}(n) \tag{4.13}$$

The result of raw outlier measure inversely depends on the average proximities. The average of these raw measures and their deviations from the average are ascertained for each one cases. The final outlier measure is obtained by subtracting

the average from every raw measure, and afterwards dividing it by absolute deviation.

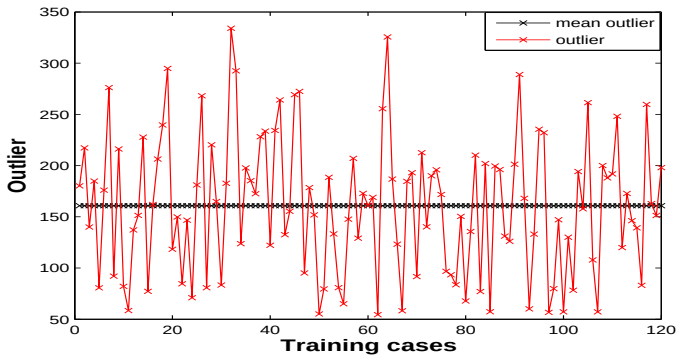
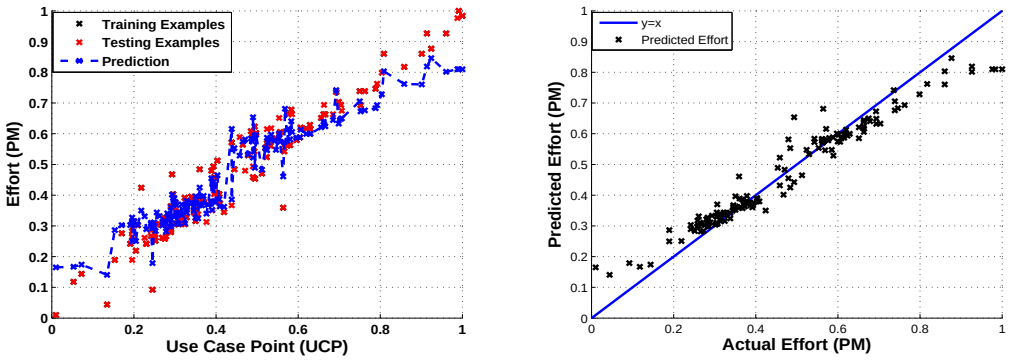


Figure 4.8: Outlier

Figure 4.8 describes the outlier value generated using random forest technique for 120 number of training cases. The outlier value is dependent on the proximity value generated using RF technique, which means that the outlier value is higher for lower proximity value and vice versa. Figure 4.8 displays the deviation of outlier value from the mean outlier. The training cases for which the outlier value is higher, will generate the predicted effort value deviated more from actual effort value. This deviation is clearly visible from Figures 4.9a and 4.9b.



(a) (b)

Figure 4.9: Random Forest Technique based Effort Estimation Model for UCP

Figures 4.9a and 4.9b display the effort estimation model obtained using RF technique. These figures show the variation of actual effort from the predicted result obtained using RF technique.

4.4.2 Model Design using Various SVR Kernel Methods

After partitioning data into learning set and validation set, the model selection for ϵ and γ is performed using 5-fold cross validation process. In this research, to perform

model selection, the ϵ and γ values are varied over a range. The γ value ranges from 2^{-7} to 2^{7} and ϵ value ranges from 0 to 5. Hence, ninety number of models are generated to perform model selection operation.

Table 4.8: Validation Errors Obtained Using SVR Linear Kernel for UCP

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	**0.0031**	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-6}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-5}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-4}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-3}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-2}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-1}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490
2^{0}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490
2^{1}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490
2^{2}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490
2^{3}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490
2^{4}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490
2^{5}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490
2^{6}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490
2^{7}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490

Table 4.9: Validation Errors Obtained Using SVR Polynomial Kernel for UCP

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.0445	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-6}	0.0445	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-5}	0.0445	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-4}	0.0443	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-3}	0.0427	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-2}	0.0318	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-1}	0.0090	0.0490	0.0490	0.0490	0.0490	0.0490
2^{0}	**0.0077**	0.0490	0.0490	0.0490	0.0490	0.0490
2^{1}	0.0089	0.0490	0.0490	0.0490	0.0490	0.0490
2^{2}	0.0090	0.0490	0.0490	0.0490	0.0490	0.0490
2^{3}	0.0090	0.0490	0.0490	0.0490	0.0490	0.0490
2^{4}	0.0090	0.0490	0.0490	0.0490	0.0490	0.0490
2^{5}	0.0090	0.0490	0.0490	0.0490	0.0490	0.0490
2^{6}	0.0086	0.0490	0.0490	0.0490	0.0490	0.0490
2^{7}	0.0308	0.0490	0.0490	0.0490	0.0490	0.0490

Tables 4.8 and 4.9 show the validation error of ninety numbers of models generated for UCP using SVR linear kernel and SVR polynomial kernel respectively based on the value of ϵ and γ for 149 numbers of project dataset. For SVR Linear kernel, *0.0031* value has been chosen as the minimum validation error. Based on the minimum validation error, the model where the result is maximum is C = 0.95588, $\gamma = 0.0078125$ and $\epsilon = 0$. For SVR Polynomial kernel, *0.0077* value has been chosen as the minimum validation error. Based on the minimum validation error, the best model is C = 0.95588, $\gamma = 1$ and $\epsilon = 0$.

Table 4.10: Validation Errors Obtained Using SVR RBF Kernel for UCP

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.0294	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-6}	0.0185	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-5}	0.0075	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-4}	0.0038	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-3}	0.0030	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-2}	0.0030	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-1}	0.0030	0.0490	0.0490	0.0490	0.0490	0.0490
2^{0}	0.0030	0.0490	0.0490	0.0490	0.0490	0.0490
2^{1}	0.0030	0.0490	0.0490	0.0490	0.0490	0.0490
2^{2}	0.0029	0.0490	0.0490	0.0490	0.0490	0.0490
2^{3}	**0.0028**	0.0490	0.0490	0.0490	0.0490	0.0490
2^{4}	0.0028	0.0490	0.0490	0.0490	0.0490	0.0490
2^{5}	0.0028	0.0490	0.0490	0.0490	0.0490	0.0490
2^{6}	0.0029	0.0490	0.0490	0.0490	0.0490	0.0490
2^{7}	0.0031	0.0490	0.0490	0.0490	0.0490	0.0490

Table 4.11: Validation Errors Obtained Using SVR Sigmoid Kernel for UCP

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.0362	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-6}	0.0294	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-5}	0.0185	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-4}	0.0075	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-3}	0.0038	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-2}	**0.0031**	0.0490	0.0490	0.0490	0.0490	0.0490
2^{-1}	0.0034	0.0490	0.0490	0.0490	0.0490	0.0490
2^{0}	0.0127	0.0490	0.0490	0.0490	0.0490	0.0490
2^{1}	0.2074	0.0490	0.0490	0.0490	0.0490	0.0490
2^{2}	3.8517	0.0490	0.0490	0.0490	0.0490	0.0490
2^{3}	16.3524	0.0490	0.0490	0.0490	0.0490	0.0490
2^{4}	7.4539	0.0490	0.0490	0.0490	0.0490	0.0490
2^{5}	2.3338	0.0490	0.0490	0.0490	0.0490	0.0490
2^{6}	1.0030	0.0490	0.0490	0.0490	0.0490	0.0490
2^{7}	0.1488	0.0490	0.0490	0.0490	0.0490	0.0490

Tables 4.10 and 4.11 show the validation error of ninety numbers of models generated for UCP using SVR RBF kernel and SVR Sigmoid kernel respectively based on the value of ϵ and γ for 149 numbers of project dataset. For SVR RBF

kernel, *0.0028* value has been chosen as the minimum validation error. Based on the minimum validation error, the model, where the result is maximum is C = 0.95588, $\gamma = 8$ and $\epsilon = 0$. For SVR Sigmoid kernel, *0.0031* value has been chosen as the minimum validation error. Based on the minimum validation error, the best model is C = 0.95588, $\gamma = 0.25$ and $\epsilon = 0$. Based on model parameters value, the model has been again trained and tested using training and testing data set respectively to estimate the effort.

The results obtained using proposed models generated using the SVR linear, polynomial, RBF and sigmoid kernel for UCP using 149 project dataset have been plotted as shown in Figures 4.10a, 4.10b, 4.10c and 4.10d respectively. These figures display the actual effort and the predicted effort obtained for UCP using the four SVR kernel methods taking into consideration of 149 project dataset.

SVR Linear Kernel Result:

Param: -s 3 -t 0 -c 0.95588 -g 0.0078125 -p 0

* Mean Squared Error (MSE_TEST) = 0.0017

* Squared correlation coefficient = 0.9557

SVR Polynomial Kernel Result:

Param: -s 3 -t 1 -c 0.95588 -g 1 -p 0

* Mean Squared Error (MSE_TEST) = 0.0059

* Squared correlation coefficient = 0.8703

SVR RBF Kernel Result:

Param: -s 3 -t 2 -c 0.95588 -g 8 -p 0

* Mean Squared Error (MSE_TEST) = 0.0011

* Squared correlation coefficient = 0.9718

SVR Sigmoid Kernel Result:

Param: -s 3 -t 3 -c 0.95588 -g 0.25 -p 0

* Mean Squared Error (MSE_TEST) = 0.0020

* Squared correlation coefficient = 0.9504

In these graphs, it is clearly shown that the data points are very little dispersed than the regression line. Hence the correlation is higher. While comparing the dispersion of data points from the predicted model in the above graphs, it is clearly visible that the data points are less dispersed for SVR RBF kernel based model than other models. Hence, this model exhibits less error values and higher prediction accuracy value. The *squared correlation coefficient*(r^2) is also known as the *coefficient of determination*. It is one of the best means for evaluating the strength of a

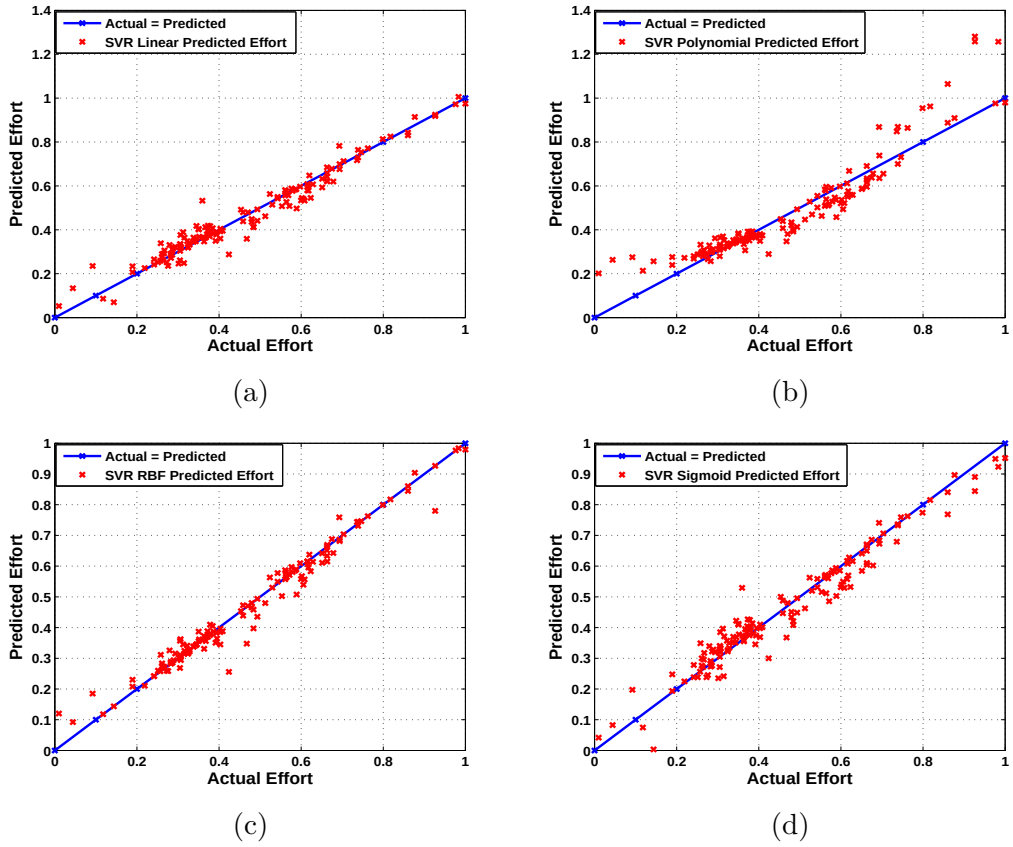


Figure 4.10: SVR Linear, Polynomial, RBF and Sigmoid Kernel based Effort Estimation Model for UCP using 149 Project Dataset

relationship. In the output generated, it is quite clearly mentioned that the *squared correlation coefficient* value for SVR Linear, RBF and Sigmoid kernel is very high (greater than 0.9). Hence it can be concluded that there is a strong positive correlation exists between the UCP and the predicted effort required to develop the software i.e., a minor change in the class point value results in significant change in the predicted effort value.

4.5 Comparative Analysis

The SGB method creates a tree ensemble, and uses the method of randomization during the creations of the trees. The prediction accuracy is calculated by feeding the result obtained from one tree to the next tree in the series. However, RF builds trees in parallel and also uses voting method on the prediction.

Table 4.12 gives a relative investigation of the outcomes acquired by a few articles specified in the related work section. The prediction accuracy (PRED) values is taken as a measure in order to evaluate the performance obtained using techniques

Table 4.12: Comparison of Prediction Accuracy Values of Related Works

Sl. No.	Related Papers	Technique Used	Prediction Accuracy
1	Issha et al. [59]	3 Novel UCP model	67%
2	Nasif et al. [66]	Treeboost Model with 84 dataset	88%
3	Nasif et al. [64]	ANN Model	90.27%
4	Nasif et al. [60]	Regression Model	95.8%

mentioned in those articles. Results indicate that, a maximum of 95% prediction accuracy is achieved using regression analysis technique for UCP. The outcomes acquired from the techniques presented in related work section is compared against the proposed methodology, which is displayed in Table 4.13. The results obtained using proposed technique shows enhancement in PRED value.

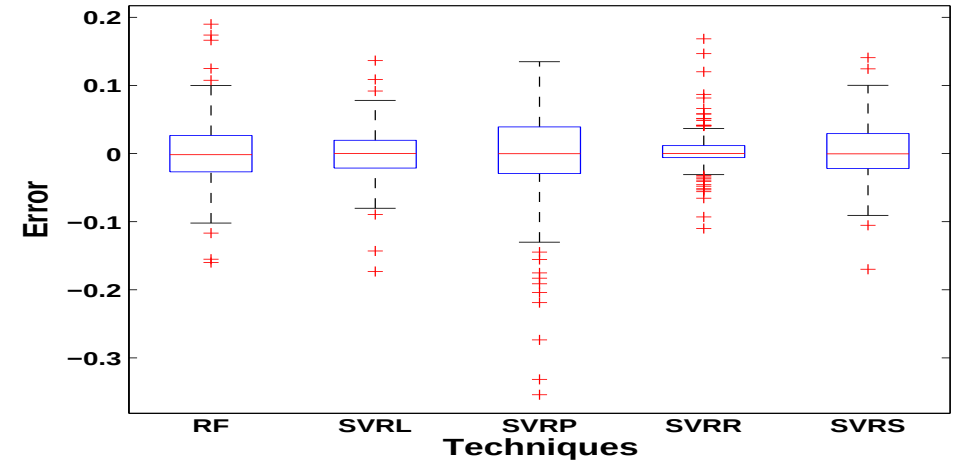
Table 4.13: Comparison of MMER and PRED Values between the Log-Linear Regression, Random Forest and Various SVR Kernel Techniques for 149 Project Dataset

	MMER	PRED(25)	PRED(50)	PRED(75)
Log-Linear Regression [24]	0.3920	37.1	75.7	94.2
Random Forest	0.0892	93.2886	97.3154	97.9866
SVR Linear Kernel	0.0876	94.6309	97.3154	97.9866
SVR Polynomial Kernel	0.1126	90.6040	96.6443	96.6443
SVR RBF Kernel	0.0549	96.6443	97.9866	97.9866
SVR Sigmoid Kernel	0.3776	92.6174	97.3154	97.3154

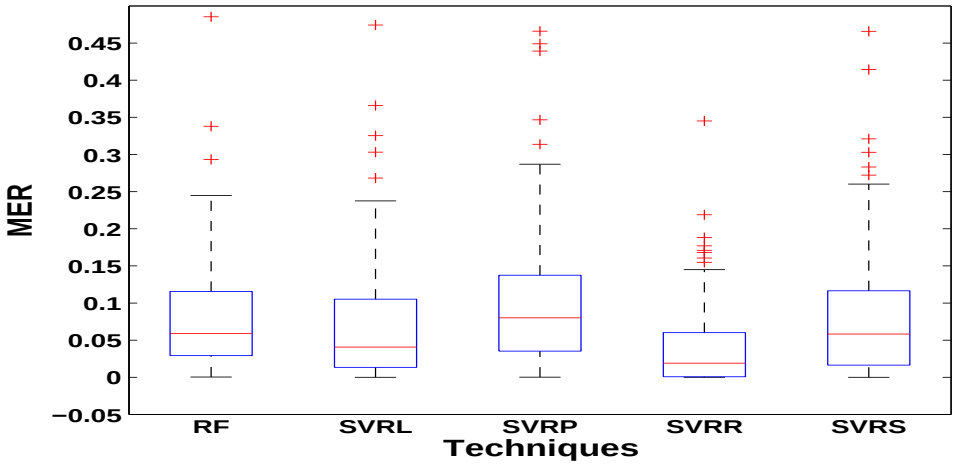
While utilizing the MMRE, and PRED performance evaluation metrics, lower value of MMRE and higher value of PRED signifies better result. Table 4.13 shows the MMRE, and PRED values acquired applying LLR, RF and various SVR kernel techniques.

Figures 4.11a and 4.11b display the box plot of Error and MER values for UCP using 149 numbers of project dataset respectively. These figures are help to illustrate the spread and differences of samples, with the help of their corresponding error values generated using RF, and different SVR kernel methods.

In order to affirm the robustness of the proposed models, the non-parametric Mann-Whitney U test [31] to calculate p-value and effect size test [131] such as Cohen's d test between diverse proposed models are processed considering absolute residuals as demonstrated in Table 4.14 for UCP using 149 project. Results demonstrate that for UCP, all the models are statistically significant at the 95% confidence internal i.e.,



(a)



(b)

Figure 4.11: Boxplot of Error and MER Values for UCP

Table 4.14: Comparison of Statistical Significance and Effect Size Test of Proposed Models for UCP using 149 Project Dataset

	Mann-Whitney p-value	**Cohen's d Effect Size**
RF vs. SVR-Linear	0.01786	0.23425
RF vs. SVR-Polynomial	0.04006	0.27873
RF vs. SVR-RBF	< 0.00001	0.15550
RF vs. SVR-Sigmoid	0.04109	0.12929

p-value < 0.05. From the results provided in Table 4.14, it is evident that the effect size is mostly small in every cases.

4.6 Summary

It is observed that in literature, a good number of methodologies have been proposed by researchers and practitioners for software effort estimation purpose. In this chapter, an attempt has been made to introduce use case point approach for softwares developed using object-oriented methodology. The parameters of use case point approach have been optimized using the RF and various SVR kernel techniques. The RF and SVR kernel techniques are ensemble learning methods for regression, which combine the results from different models of similar type or different and gives result which is usually better than the result obtained from other individual models. The RBF kernel uses more complex decision boundary. The results are based on non-linear data and the RBF kernel makes a good default kernel, if the problem applies on non-linear model. After analyzing the results, it is observed that effort estimation model developed on the considered dataset using SVR RBF kernel technique help to yield better quality results than other machine learning techniques based effort estimation models.

Chapter 5

Effectiveness of Machine Learning Techniques for Effort Estimation of Web-based Applications

5.1 Introduction

The requirements for development of web-based applications are very often complex by nature. According to the Molokken and Jorgensen report [48], about 30-40% extra effort in terms of man month is often spent in software development on an average. So, the analysts are very much conscious at present to avoid any amount of extra overhead. As per Donal J. Reifer [9], effort estimation models, which are helpful for conventional software development, are not extremely precise for effort estimation of web-based software development. Traditional software size and effort estimation techniques are not adequate to capture specific features of the development, which can influence the size and effort required in the development of web applications [140]. Different approaches are proposed by various authors in the literature on estimation of web-based applications. Broadly, there are generally two approaches for sizing web applications [141], i.e., Lines of Code (LOC) and Functional Point Analysis (FPA). There are also some other custom solutions as illustrated by Azhar et al. [142].

Functional Size Measurement (FSM) is a concept on step by step instructions used to evaluate the software size in terms of functional requirements requested by a user. The initial estimation method developed to support this concept was Function Point Analysis (FPA) proposed by Allan Albrecht in 1979 [38]. Albrecht defined a Function Point (FP) as a unit of measure that represents the amount of business functionalities of an information system, provided to a client. The advantage of these methods lies in the fact that they are independent of technology or programming language used and can be used through the entire development life cycle [143]. With FPA method, the size of a software application and, the development effort of the software application

at the beginning of the development process can be estimated, which might not be the case of other methods. Costagliola et al. [140] have adopted FP count in web applications.

The research work carried out in this thesis deals with the formal estimation using different machine learning (ML) techniques such as Decision Tree (DT), Stochastic Gradient Boosting (SGB), Random Forest (RF) and Support Vector Regression (SVR) Kernels for development of effort estimation models by following a function point approach. Supervised machine learning is the quest for algorithms that requires training dataset to convey general hypotheses and predict the future instances [144]. The supervised machine learning algorithm looks to construct a model that can predict the response values obtained from training dataset for another dataset. The basic objective of test dataset is to validate the trained model. Models with higher prediction accuracy can be achieved by utilizing larger training dataset, which can also be generalized for new datasets [145, 146]. The ML techniques are accredited for their capacity to create more qualitative results when managing issues where there exists complex connections in the middle of inputs and yields, and where there is a distortion in the inputs by high noise levels. The validation of the proposed models were carried out with the help of the ISBSG Release 12 [147] dataset. The function data collected from ISBSG Release 12 are then used as input to machine learning techniques-based effort estimation models in order to predict the effort and assess their performance.

5.2 Dataset Description

The ISBSG dataset, Release 12 [147] is used in this study for developing effort estimation models for web-based software. The ISBSG Release 12 dataset has details of 6006 number of projects, out of which 936 number of projects are based on web projects. There are mainly three categories of data development approach, i.e., new development, enhancement and redevelopment ones. Out of which, only 18 records are available for redevelopment type of web projects [148]. The accuracy of the results obtained using this category of projects can not be guaranteed, due to lack of large number of dataset. Hence, in this proposed approach, redevelopment type of web projects are not taken into consideration for implementation purpose. The dataset is very heterogeneous in nature, and the productivity (ratio between software effort and software size) varies significantly even with the same size metric [88]. For instance, for projects of same metric size IFPUG, the value of productivity varies between 0.2 and 257.8. For example, if a project size is of 100 units, the effort required to develop this project varies between 20 hours (if productivity is 0.2) and 25780 hours

(if productivity is 257.8). This is a concern, which needs to be addressed. To solve this issue, three different subsets were taken into consideration for both the new and enhancement types of web projects separately, based on the value of the productivity. The first subset is chosen, when the values of productivity vary between 0 and 6.9 (both inclusive). The second one is chosen, when the productivity values are between 7 and 14.9, and the third one is chosen, when productivity values are greater than or equal to 15. The statistical profile of three sub-categories of project's data collected from ISBSG Release 12 dataset is depicted in Table 5.1. In this table, the three categorization of new development type of projects are denoted as New Dataset 1, 2 and 3. Similarly, the three categorizations of enhancement type of projects are denoted as Enhance Dataset 1, 2 and 3.

Table 5.1: Statistical Profile of ISBSG Release 12 Dataset for Web-based Applications

Project Type	Minimum	Maximum	Mean	Median	Std. Dev.	Skewness	Kurtosis
New Dataset 1	83	29610	1699.91	750	3211.59	5.72	42.73
New Dataset 2	255	18314	3899.45	2610	3686.37	1.95	3.53
New Dataset 3	540	60826	8626.47	5299	10932.6	3.07	10.45
Enhance Dataset 1	8	7836	625.38	301	1021.93	4.26	22.95
Enhance Dataset 2	28	17400	1549.38	950	2225.31	4.35	23.40
Enhance Dataset 3	149	47493	4826.30	2808	6523.35	3.65	18.46

This study intends to apply different machine learning techniques to estimate the effort for developing new and enhanced web projects using the IFPUG Function Points approach. The set of useful ones (considered for analysis and filtering for use in the models developed) are described below:

- Rating: In this category, the following attributes are considered to be important:
  - Data Quality Rating: The ISBSG quality reviewers have assigned a rating code of A, B, C or D to the project data denoting it's extent of integrity, with A being the best and D being the worst.
  - Unadjusted Function Point Rating: The ISBSG quality reviewers have assigned a rating code of A, B, C or D to the Functional Size data denoting it's extent of integrity, with A being the best and D being the worst.
- Major Grouping Attributes: In this category, the count approach attribute and the development type attribute are considered to be important. The former describes the methods used to estimate the size of the project. Mostly, the Functional Size Measurement Method is used (IFPUG, MARK II, NESMA, FiSMA, COSMIC-FFP etc., but the major portion of data considered is from IFPUG data). The latter describes whether the development was a new one or an enhancement or a re-development.

- Sizing attributes: In this category, the Adjusted Function Points attribute is considered to be important. For IFPUG, NESMA, FiSMA and MARK II counts, this is the adjusted size (adjusted by VAF value).

- Effort attributes: In this category, Normalized Work Effort attribute and the Summary Work Effort attribute are considered to be important. The former is the effort value of the full development cycle whereas the latter gives the total effort in number of hours for each project. For some projects, the full development cycle time is not covered and Normalized Work Effort attribute is an approximation for full cycle effort. In other cases, where the full development cycle time is covered, the Normalized Work Effort and Summary Work Effort are same.

- Productivity attributes: In this category, Normalized Productivity Delivery Rate attribute and the Pre 2002 Productivity Delivery Rate attribute are considered to be important. The former implies the conveyance rate of the projects utilized and reported subsequent to the year 2002. This productivity conveyance rate in hours per useful size unit is figured from Normalized Work Effort partitioned by the Functional Size (UFP count) of the full improvement cycle; while the latter implies the total effort in number of hours for every project. This productivity conveyance rate in hours per functional size unit is ascertained from Summary Work Effort partitioned by AFP count.

- Architecture: In this category, the web Development attribute is considered to be important. It indicates whether the entry is for web-based project or not.

- Size attributes: In this category, the files associated with Function Point Categories such as total no. of Input, Output, Enquiry, File, Interface etc. are considered to be important and taken into consideration.

- Size Other than FSM: Lines of Code, Lines of Code not Statements etc. are provided.

5.3 Proposed Work

The proposed approach is implemented using the ISBSG dataset. Figure 5.1, demonstrates the steps carried out in the proposed research work applied to compute the effort required to develop web-based applications using different machine learning techniques.

The steps taken to determine the effort of a software product are described below:

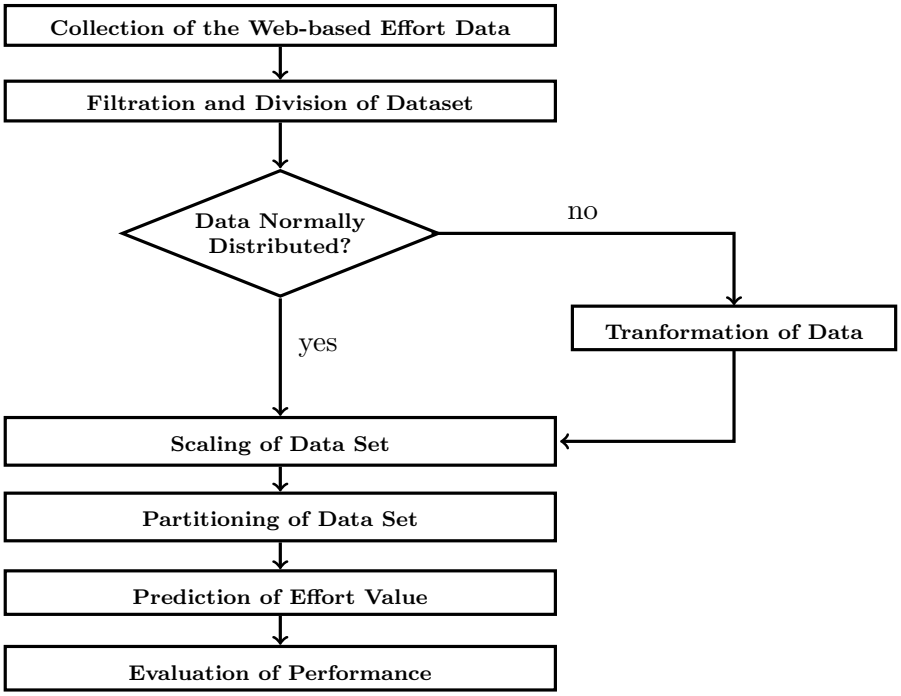


Figure 5.1: Steps Followed for Effort Estimation of Web-based Applications using Various Machine Learning Techniques

1. **Collection of the Web-based Effort Data:** The data used for developing effort estimation models is ISBSG Release 12 data. It is obtained from the ISBSG community.

2. **Filtration and Division of Dataset:** The ISBSG data is filtered by using the following attributes:

   - Web Architecture: Only web-based applications and web projects are taken into consideration. Hence, from 6006 number of projects, 936 number of projects are selected.
   - Data Quality Rating: Only projects with a data quality rating of A and B are taken into consideration.
   - Unadjusted Function Point Rating: Only projects with an unadjusted function point rating of A, B and C are taken into consideration.

   After the requisite data are filtered, the development type attribute is used for separating data rows into three major groups:

   - New developments
   - Enhancement Projects

- Re-developments

3. **Data Normally Distributed?**: The statistical analysis of the collected dataset has been performed. It is verified that whether the collected dataset follows normal distribution or not based on the values of skewness and kurtosis. If data are normally distributed, then it will directly proceed to the data normalization step. Otherwise, the data need to be transformed so as to make it normally distributed.

4. **Transformation of Data**: If the dataset is not normally distributed, then the logarithmic transformation has been applied over the dataset to make it normally distributed. Histograms using method plots are drawn as as to verify the distribution of data before and after transformation.

5. **Scaling of Dataset:** This step deals with generating the scaled values of the input vectors individually with in the range [0,1]. Let us consider Y as complete dataset and y as an element of the dataset, then normalized value of y is calculated as:

$$y' = \frac{y - \min(Y)}{\max(Y) - \min(Y)} \tag{5.1}$$

   where y' = Normalized value of y within range [0,1], $\min(Y)$ = min. value of Y and $\max(Y)$ = max. value of Y. When $\max(Y) = \min(Y)$, $y' = 0.5$.

6. **Partition of Dataset:** The whole dataset is apportioned into training and test set. The training set is utilized for model estimation, though the test set is utilized just for assessing the anticipated effort of the final model. This stride is executed utilizing 10-FOLD cross validation process.

7. **Prediction of Effort Value:** The effort value is predicted using various ML techniques used in this study i.e., DT, SGB and RF. The detailed description of the steps carried out for predicting the effort using these machine learning techniques are provided in the Experimental Details section.

8. **Evaluation of Performance:** The assessment of the different ML techniques-based effort estimation models is carried out by considering the RMSE, MAE, MMER and PRED(x) results obtained from test set in order to evaluate their performance. The model giving lower values of RMSE, MAE, MMER and higher values of PRED(x) is considered as the acceptable model.

The results obtained using the above models are compared to assess the performance. Comparative analysis with the results of models considered by various

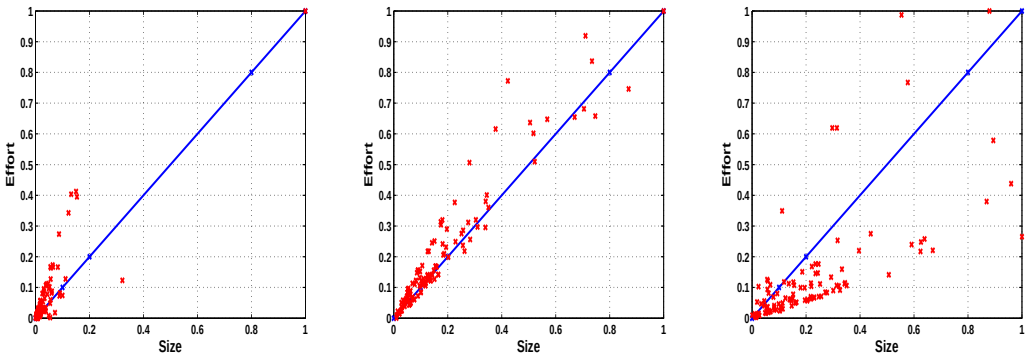
authors in literature is also carried out in order to verify if the proposed work helps in obtaining improved prediction accuracy or not.

5.4 Experimental details

After processing the input, the following number of projects data for each type are obtained.

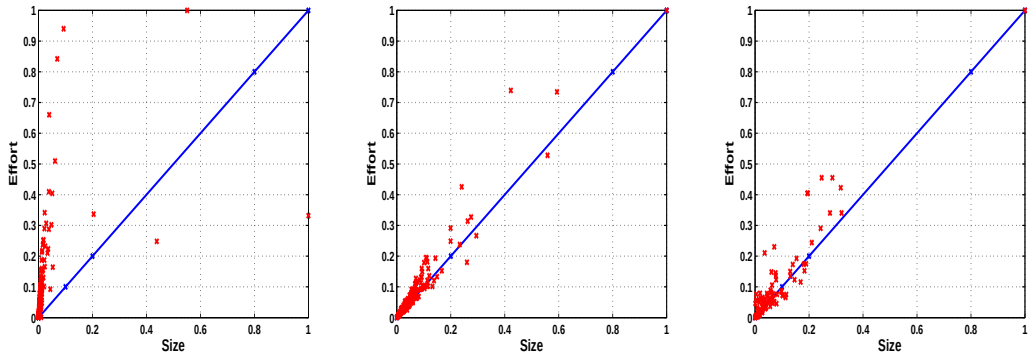
- For New development type web projects: 368 records.
  - Dataset 1: 140 records
  - Dataset 2: 124 records
  - Dataset 3: 104 records
- Enhancement type web projects: 511 records.
  - Dataset 1: 247 records
  - Dataset 2: 163 records
  - Dataset 3: 101 records

Figures 5.2a, 5.2b and 5.2c depict the relationship between software size (adjusted function points) and software effort (person-hours) in each of the three datasets for new web projects. Similarly, Figures 5.3a, 5.3b and 5.3c depict the relationship between software size (adjusted function points) and software effort (person-hours) in each of the three datasets for enhanced web projects.



(a) (b) (c)

Figure 5.2: Software Size (AFP) vs. Effort Graph based on Dataset 1, Dataset 2 and Dataset 3 for New Web Projects



(a) (b) (c)

Figure 5.3: Software Size (AFP) vs. Effort Graph based on Dataset 1, Dataset 2 and Dataset 3 for Enhanced Web Projects

From these figures, it is observed that the enhance dataset 1 contains more number of outliers than other datasets. From Table 5.1, it has been observed that the new datasets 2 and 3 are more normally distributed, based on the values of the skewness and kurtosis than other datasets. Adjusted Function Points(AFP) and Resource level attributes are considered as input and Normalized Work Effort is considered as output of the effort estimation models. The reasons behind this are given below:

- The final stage of estimation in most of the Functional Size Measurement methods such as IFPUG, NESMA, and MARK-II calculates the effort from the AFP value. This value is available in ISBSG Release 12 dataset; whereas using the other attributes such as ILF, EIF, DET RET, EI, EO, VAF etc. from initial phases of counting, the effort calculation is not possible. This is because these values are not individually provided in the dataset. Hence, after finding AFP value, it is then used for calculating the final effort. So, AFP is taken as the input.

- Apart from AFP value, productivity rate of projects and Resource level are also playing a major role while calculating final project effort value. In this study, AFP and resource level are the attributes taken as input arguments to different machine learning models for calculating predicted effort.

- Normalized Work Effort is the effort value of the full development cycle; whereas the later presents the total effort computed in terms of person-hours documented against the project. For some projects, the full development cycle time is not covered and Normalized Work Effort attribute is an approximation for full cycle effort. In other cases, where the full development cycle time is covered, the normalized Work Effort and Summary Work Effort values are same. The data

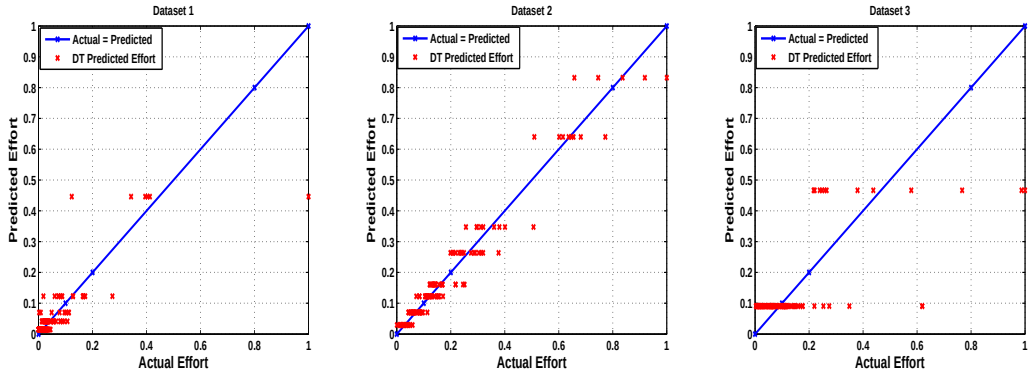
about whether a project has completed during the full developmental cycle time or not, is not provided in the ISBSG dataset.

5.4.1 Model design using Decision Tree Technique

The parameters of the DT model were chosen so that the error is minimal with one exception where the tree is pruned. The tree was pruned based on the minimum value of the cross validation error. Pruning helps to simplify the model; but it has also a negative impact on the value of accuracy. The parameters of the DT model are as follows:

- Min. Number of Rows in a Node: 5
- Min. Node Size for Splitting: 10
- Max. Number of Levels in a Tree: 10
- Smooth Minimum Spikes: 3

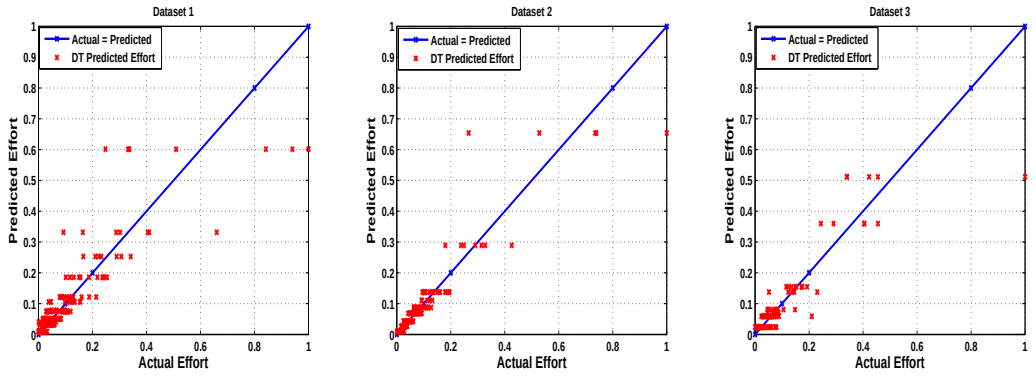
These parameters value are decided by picking proper combinations in order to produce results with maximum accuracy considering DT-based model for estimating web development effort.



(a) (b) (c)

Figure 5.4: Actual vs. Predicted Effort Graph using DT Technique based on Dataset 1, Dataset 2 and Dataset 3 for New Web Projects

Figures 5.4a, 5.4b and 5.4c depict the variation of predicted effort values from actual by applying DT-based effort estimation model for new web projects. Similarly, Figures 5.5a, 5.5b and 5.5c depict the variation of predicted effort values from actual by applying DT-based effort estimation model for enhanced web projects. From these figures, it is observed that there is very less deviation between the predicted effort and the actual effort values for new dataset 2 and enhanced dataset 2. But this is little bit on a higher side for rest of the type of new and enhanced web projects.



(a) (b) (c)

Figure 5.5: Actual vs. Predicted Effort Graph using DT Technique based on Dataset 1, Dataset 2 and Dataset 3 for Enhanced Web Projects

5.4.2 Model design using Stochastic Gradient Boosting Technique

To outline a SGB technique based effort estimation model, the accompanying steps are utilized.

1. The coefficient of G_0 is assessed by computing the mean of the actual effort value i.e., the target predictor.

2. By taking the help of stochastic factor to nourish the next tree, an arbitrary percentage of rows are chosen. If the value is fixed as 0.5, then 50% of rows are arbitrarily picked.

3. The residuals obtained from each row are sorted and after that the residues utilizing the Huber's Quantile Cutoff factor are transfered, which is termed as *pseudo-residuals*.

4. The principal tree (S1) is fitted to the pseudo-residuals.

5. For each of the terminal nodes, the nodes, which anticipated estimations are computed utilizing the mean of the pseudo-residuals.

6. The residuals between the anticipated estimations and the pseudo-residuals that nourished the tree are figured.

7. The Huber's Quantile Cutoff is enforced again over the result obtained from step 6 and then processed the mean residuals.

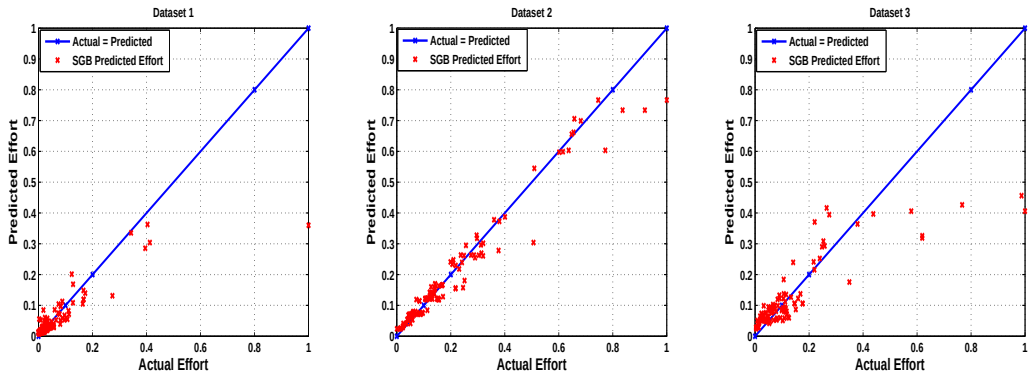
8. By computing the distinction between the mean of the anticipated estimations of the tree and mean residual, the boost coefficient (A1) of the tree is obtained.

9. At the end, in order to impede the learning procedure, the shrinkage factor is multiplied with the boost coefficient.

The accompanying parameters help to find the predicted effort utilizing the SGB technique.

- Number of Trees: 1000
- Depth of Individual Tree: 5
- Huber's Quantile Cut off : 0.95
- Min. Node Size for Splitting: 10
- Influence Trimming Factor : 0.01
- Shrinkage Factor : 0.05
- Smooth Minimum Spikes: 5
- Stochastic Factor : 0.5

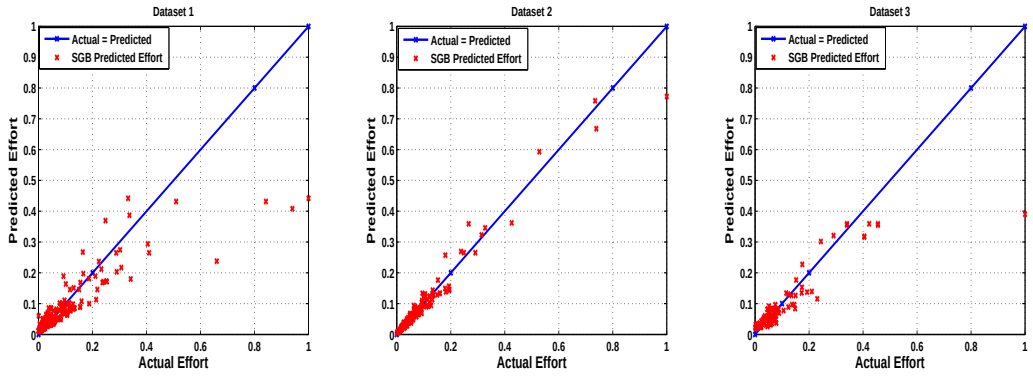
The comprehensive depiction of these parameters were already presented in Section 1.4.2. The values of these parameters are decided by picking proper combinations in order to produce results with maximum accuracy considering SGB-based model for estimating web development effort.



(a) (b) (c)

Figure 5.6: Actual vs. Predicted Effort Graph using SGB Technique based on Dataset 1, Dataset 2 and Dataset 3 for New Web Projects

Figures 5.6a, 5.6b and 5.6c depict the variation of values indicating predicted effort from actual by applying SGB-based effort estimation model for new web projects. Similarly, Figures 5.7a, 5.7b and 5.7c depict the variation of predicted effort values



(a) (b) (c)

Figure 5.7: Actual vs. Predicted Effort Graph using SGB Technique based on Dataset 1, Dataset 2 and Dataset 3 for Enhanced Web Projects

from actual by applying SGB-based effort estimation model for enhanced web projects. From these figures, it is observed that for the new web projects especially for New Dataset 2 and New Dataset 3, there is marginally very less deviation between the predicted effort and the actual effort values. Hence the accuracy is very high. But this is little bit on a higher side for other type of new and enhanced web projects.

5.4.3 Model design using Random Forest Technique

After obtaining the normalized values of the input parameter, an arbitrary random vector is selected having randomness in input data and with splitting at nodes. Hence, initially an arbitrary random vector is selected to provide randomness in input data and start the implementation process. The total number of data are divided into two subsets i.e., training set and test set using the above arbitrary. Prediction results vary according to random vector. So an evaluation function (1- MMER + Prediction Accuracy) has been developed to find a random vector. The random vector, which provides optimum value for the evaluation function is considered as final random vector. The Brieman's algorithm is popularly used to implement the random forest technique [19]. In order to obtain a random forest technique-based effort estimation model, the steps presented underneath are taken into consideration. These proposed steps help in constructing each tree, while using random forest technique.

Steps of Proposed Algorithm:

1. Let F be the number of trees in the forest. A Dataset of D points having $(x_1, y_1)(x_2, y_2)....(x_D, y_D)$ is considered.

2. Each tree of the forest should be grown. Hence steps from i to vii should be repeated f times to create F number of trees.

i. Let N be the no. of training cases, and M be the no. of variables in the classifier.

ii. To select training set for the tree, a random sample of n cases - yet with substitution, from the original data of all N accessible training cases is chosen. Whatever is left of the cases, they are utilized to evaluate the error of the tree, by foreseeing their classes.

iii. A RF tree 'T_f' is developed for the loaded data, by repeatedly rehashing the accompanying steps for every terminal node of the tree, till the minimum node size n_{min} is arrived. Keeping in mind the end goal to make more randomness, distinctive dataset for each one tree is made.

iv. The no. of input variables 'm' is selected to discover the choice at a tree node. The value of 'm' ought to be substantially short of the value of/ 'M'.

v. For each tree node, 'm' number of variables should be randomly chosen on which the decision at that node is based.

vi. The best split focused around these 'm' variables in the training set is calculated. The value of 'm' ought to be held consistent throughout the development of the forest. Each tree should be fully grown and not pruned.

vii. Then, the results of ensemble of trees $T_1, T_2, ..., T_f, ..., T_F$ are collected.

3. The input vector should be put down for each of the trees in the forest. In regression, it is the average of the individual tree predictions.

$$Y^F(x) = 1/F \sum_{f=1}^{F} T_f(x) \tag{5.2}$$

where

$Y^F(x)$ is the predicted value for the input vector x.

$T_1(x), T_2(x), ..., T_f(x)$ represents prediction value of individual trees.

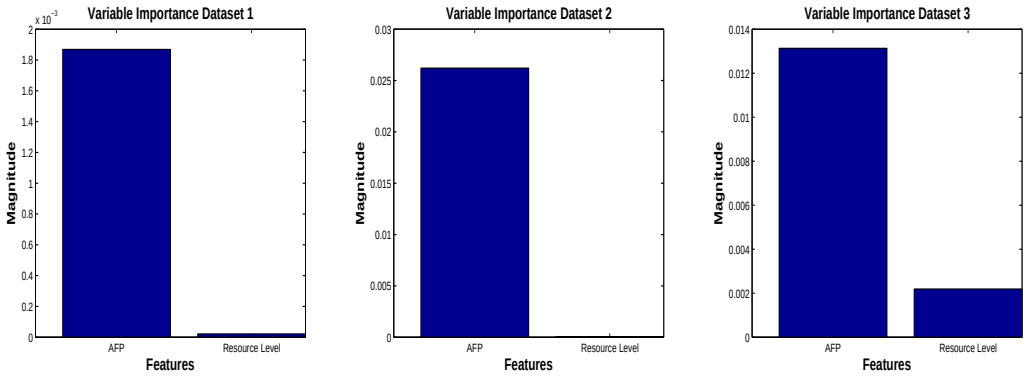
The parameters of the RF model are considered as follows:

- Number of Trees in the Forest: 500
- Min. Size Node for Splitting: 10
- Max. Number of Levels in a Tree: 10
- No. of Predictors Sampled for Splitting at Each Node: 1

There are various data objects generated by random forest technique, which need to be considered while implementing random forest technique for software effort estimation purpose. The results obtained from these data objects need to be evaluated in order to assess the performance achieved using random forest technique.

Variable Importance

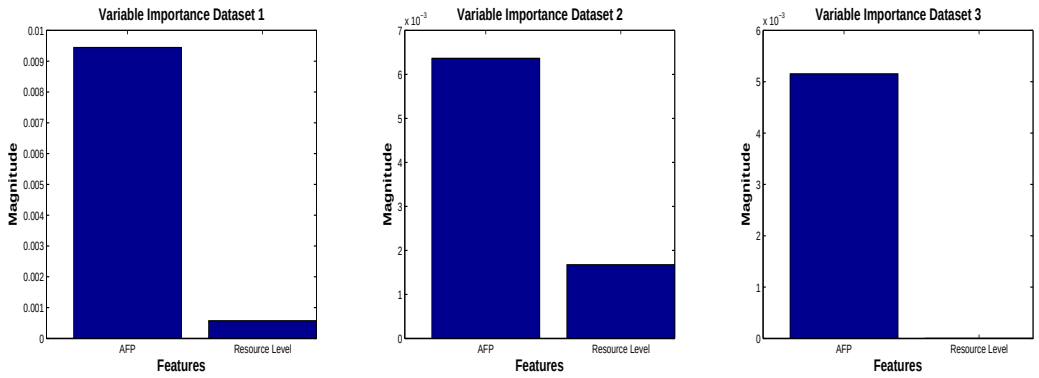
The variable importance defines the contribution of a variable in achieving prediction to a certain degree of accuracy. It is calculated by taking into consideration of its interaction with other variables. The error rate for each tree T, is calculated using the Out-of-Bag(OOB) data. Then, the permutation result of the OOB values is calculated for each variable 'v' and the error value is calculated using each tree. If the number of variables for implementing RF technique is very large, forests can be made to run once with all the variables. Then, by using only the most important variable from the initial run, forests can be run again to calculate the final predicted effort.



(a) (b) (c)

Figure 5.8: Actual vs. Predicted Effort Graph using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for New Web Projects

Figures 5.8a, 5.8b and 5.8c display the importance of three variables i.e., Adjusted Function Point (AFP), Productivity and Resource Level taken as input to the model for calculating the effort using random forest technique based on Dataset 1, Dataset 2 and Dataset 3 for new web projects respectively. Similarly, Figures 5.9a, 5.9b and 5.9c display the importance of three variables i.e., Adjusted Function Point (AFP), Productivity and Resource Level taken as input to the model for calculating the effort using random forest technique based on Dataset 1, Dataset 2 and Dataset 3 for enhanced web projects respectively. The first column in the figure represents the importance of AFP, the second column represents the importance of productivity and the third column represents the importance of resource level of the software on the effort estimation process. From these figures, it is observed that for all cases, the



(a) (b) (c)

Figure 5.9: Actual vs. Predicted Effort Graph using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for Enhanced Web Projects

impact of the AFP is highest for predicting the effort required to develop the software than productivity and resource level. The resource level variable has very less impact on the overall estimation process in case of new and enhanced type web projects, even it is on negative side for enhanced dataset 3.

The Out-Of-Bag (OOB) Error Estimate

The training set for a tree is produced by testing with substitution. During this process, one-third of the cases are left out of the sample. These cases are considered as out-of-bag (OOB) data. It helps in getting an impartial evaluation of the regression error value as the forest develops. OOB data also helps in getting estimation of variable importance. In RF, as the OOB is calculated internally during the run, cross validation of data or a different test set to obtain an impartial evaluation of the test error is not required. The computation procedure for OOB is explained below:

- During construction of each tree, an alternate bootstrap sample from the original data is used. Something like one-third of the cases from the bootstrap sample are left out and not used in the tree construction process. Hence, out of one hundred twenty data, eighty data are used in the tree construction process and rest forty data are used for testing the result.

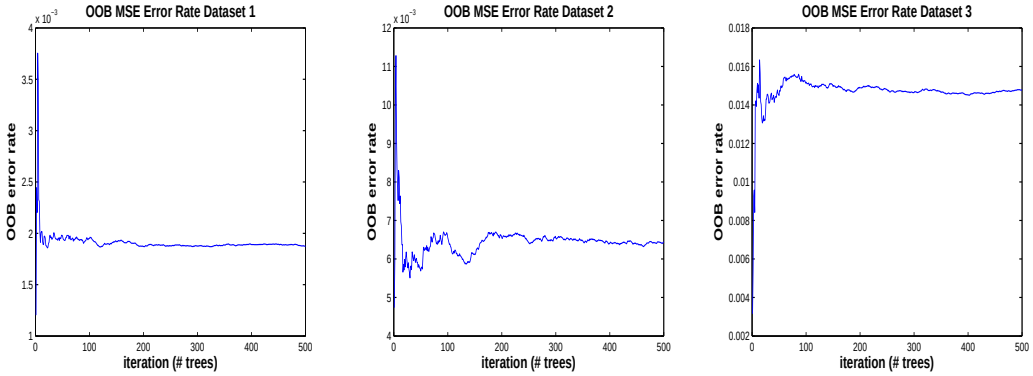
- These OOB samples are put down the kth tree to obtain a regression. Using this process, a test set is acquired for each case.

- At the end, suppose j be the predicted value which is acquired by computing the average prediction value of forest, for each time the case n was OOB. The extent of times j is not equivalent to the actual value of n averaged over all cases, is called as the *out-of-bag error estimate*.

The RF prediction accuracy can be determined from these OOB data by using the following formula:

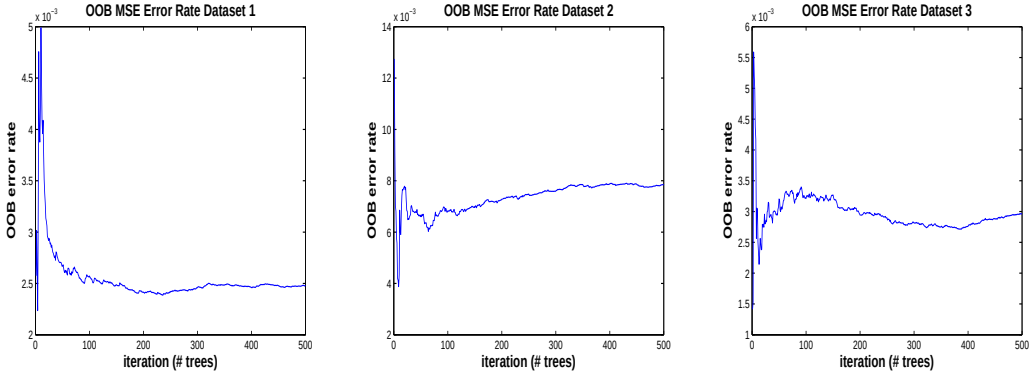
$$OOB - MSE = \frac{1}{F}\sum_{i=1}^{F}(y_i - \bar{y}_{iOOB})^2 \tag{5.3}$$

where $\bar{y}_{iOOB}$ represents the average prediction value of ith observation from all trees for which this observation has been OOB. F denotes the no. of trees in the forest and y_i represents the actual value.



(a) (b) (c)

Figure 5.10: OOB MSE Error Rate using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for New Web Projects

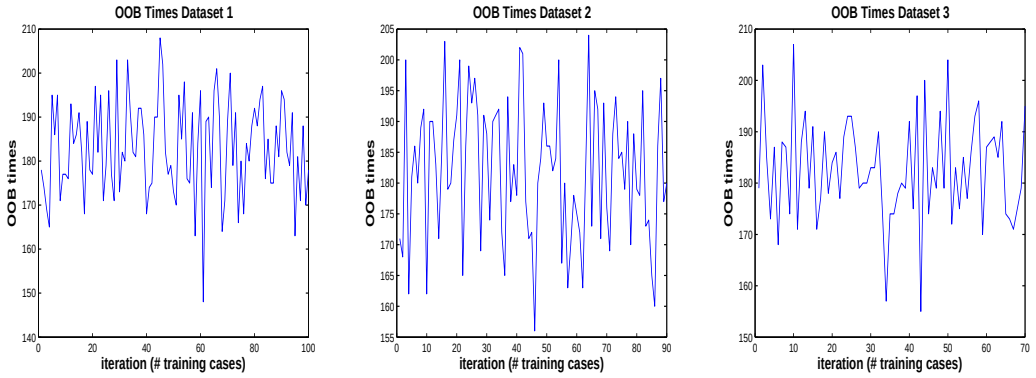


(a) (b) (c)

Figure 5.11: OOB MSE Error Rate using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for Enhanced Web Projects

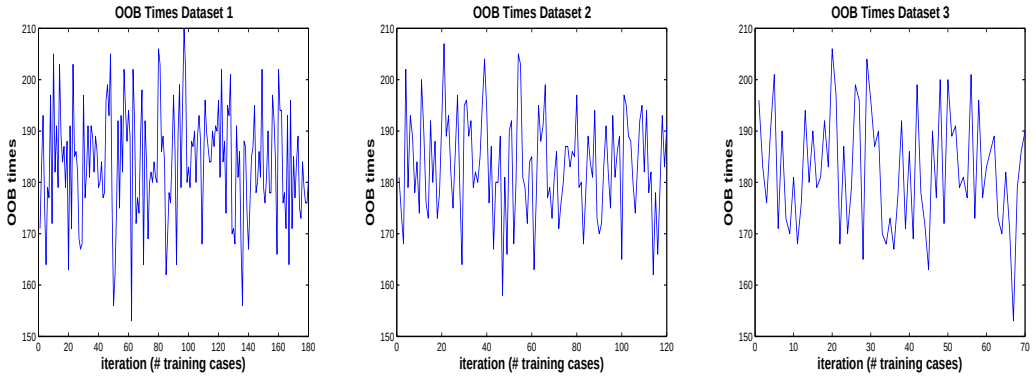
Figures 5.10a, 5.10b and 5.10c display the OOB error rate obtained for different number of trees used in the forest using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for new web projects. Similarly, Figures 5.11a, 5.11b and 5.11c display the OOB error rate obtained for different number of trees used in the forest

using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for enhanced web projects. From these figures, it is observed that during initial phase (while the number of trees used are less), the OOB error rate obtained is maximum. At the same time, steadily with the increment of the amount of trees utilized within the forest, the OOB error rate converges to minimum value. After some period, OOB error rate remains constant.



(a) (b) (c)

Figure 5.12: Number of Times Out Of Bag Occurs using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for New Web Projects



(a) (b) (c)

Figure 5.13: Number of Times Out Of Bag Occurs using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for New Web Projects

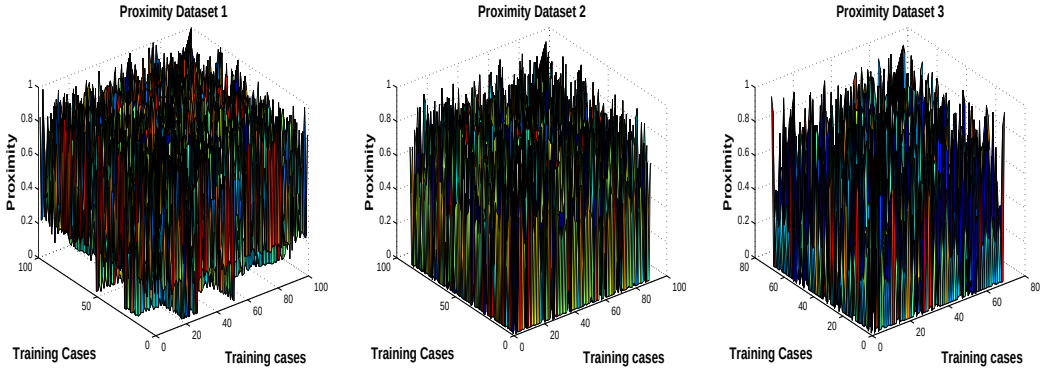
Figures 5.12a, 5.12b and 5.12c display the number of times, cases are out of bag for all training attributes using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for new web projects. Similarly, Figures 5.13a, 5.13b and 5.13c display the number of times, cases are out of bag for all training attributes using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for enhanced web projects. 100, 90 and 70 number of training attributes are used for new dataset 1, new dataset 2 and new

dataset 3 respectively. Similarly, 180, 120 and 70 number of training attributes are used for enhanced dataset 1, enhanced dataset 2 and enhanced dataset 3 respectively.

Proximities

Proximity is one of the important data objects while calculating effort using RF technique. It measures the frequency of ending up the unique pairs of training samples in the same terminal node. It also helps in filling up the missing data in the dataset and calculating number of outliers.

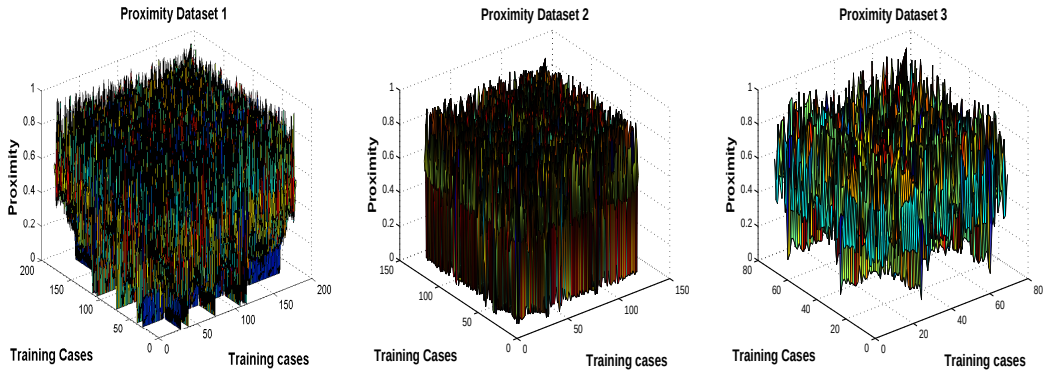
Originally, a '$N \times N$' matrix is formed by the proximities. Once a tree is developed, all the data i.e., training data and out-of-bag data are put down the tree. Its proximities should be increased by one, if it is found that two cases are in the same terminal node. Finally, the normalized values of the proximities are obtained by dividing with the number of trees.



(a) (b) (c)

Figure 5.14: Proximity using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for New Web Projects

Figures 5.14a, 5.14b and 5.14c indicate the proximity value generated using random forest technique on Dataset 1, Dataset 2 and Dataset 3 for new web projects. Similarly, Figures 5.15a, 5.15b and 5.15c describe the proximity value generated using random forest technique on Dataset 1, Dataset 2 and Dataset 3 for enhanced web projects. Three different matrices of sizes such as 100 × 100, 90 × 90 and 70 × 70, are used for generating the proximity graph for new dataset 1, new dataset 2 and new dataset 3 respectively. Similarly, matrices of size 180 × 180, 120 × 120 and 70 × 70, are used for generating the proximity graph for enhanced dataset 1, enhanced dataset 2 and enhanced dataset 3 respectively. From these figures, it is observed that, for diagonal elements, the proximity value is maximum (equals to one). But for all other elements, the proximity value is less than one. The symmetric portion adjacent to diagonal area represents other elements proximity values.



(a) (b) (c)

Figure 5.15: Proximity using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for Enhanced Web Projects

Complexity

In the proposed approach, 1000 number of trees are taken into consideration for implementing RF technique. In the usual tree growing algorithm, all descriptors are tested for their splitting performance at each node; while Random Forest only tests m try of the descriptors. Since m try is typically very small, the search is very fast.

In order to obtain the right model complexity for optimal prediction strength, pruning is usually done via cross validation for a single decision tree. This process can take up a significant portion of the computations. RF, on the other hand, does not perform any pruning at all. It is observed that in cases where there are an excessively large number of descriptors, RF can be trained in less time than a single decision tree. Hence, the RF algorithm can be very efficient.

Outlier

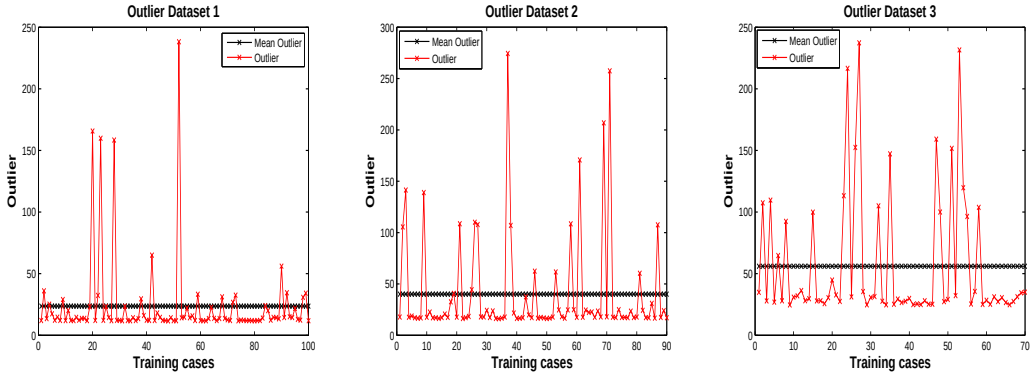
The cases that are expelled from the principal group of data and whose proximities to all different cases in the data being mostly small are defined as *Outliers*. The concept of outliers can be revised by defining outliers relative to corresponding cases. Hence, an outlier is a case whose proximities to all different cases are little. The average proximity is specified as:

$$\bar{P}(n) = \sum_{1}^{N} prox^2(n,k) \tag{5.4}$$

where 'n' and 'k' denote a training case in the regression and N represents the total no. of training cases in the forest. The raw outlier measure for case 'n' is specified as:

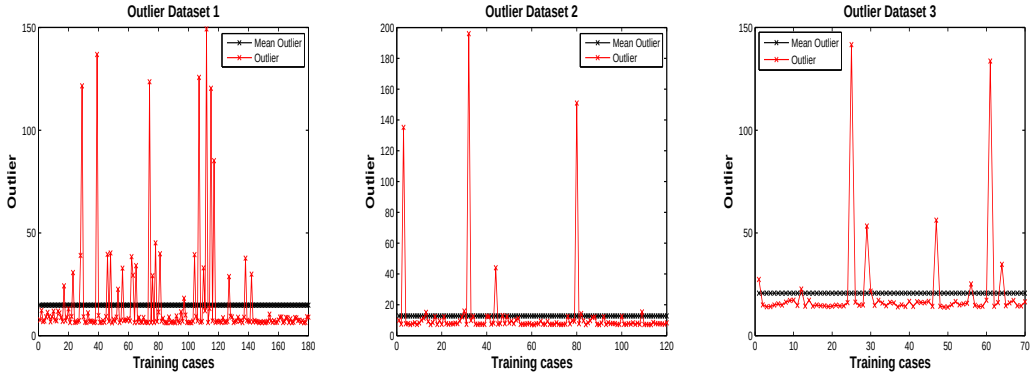
$$nsample/\bar{P}(n) \tag{5.5}$$

The result of raw outlier measure inversely depends on the average proximities. The average of these raw measures and their deviations from the average are ascertained for each case. The final outlier measure is obtained by subtracting the average from every raw measure, and afterwards dividing it by absolute deviation.



(a) (b) (c)

Figure 5.16: Outlier using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for New Web Projects

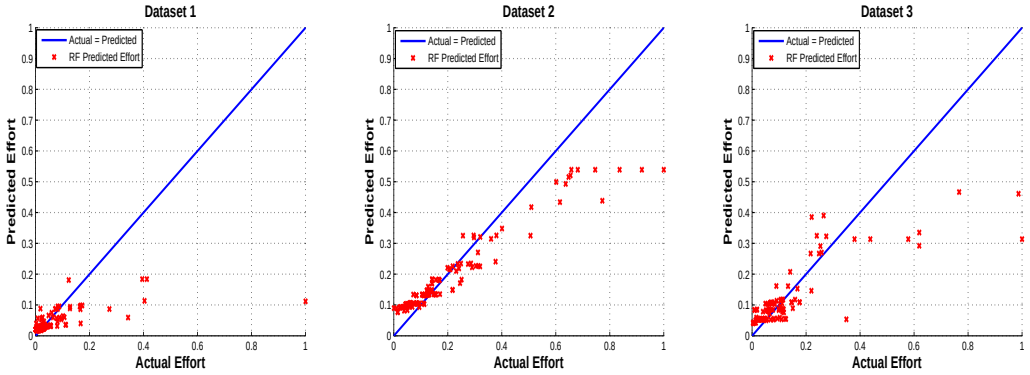


(a) (b) (c)

Figure 5.17: Outlier using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for Enhanced Web Projects

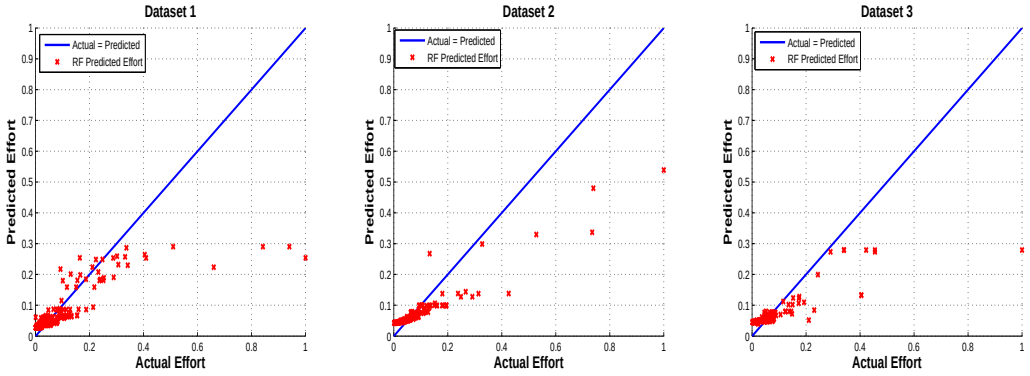
Figures 5.16a, 5.16b and 5.16c describe the outlier value generated using random forest technique with 100, 90 and 70 number of training cases on new dataset 1, new dataset 2 and new dataset 3 respectively. Similarly, Figures 5.17a, 5.17b and 5.17c describe the outlier value generated using random forest technique with 180, 120 and 70 number of training cases on enhanced dataset 1, enhanced dataset 2 and enhanced dataset 3 respectively. The outlier value is observed to be dependent on the proximity value generated using RF technique, which means that the outlier value is higher for lower proximity value and vice versa. These figures display the deviation of outlier

value from the mean outlier. The training cases for which the outlier value is higher, will generate the predicted effort value deviated more from actual effort value. This deviation is clearly visible from model design figures.



(a) (b) (c)

Figure 5.18: Actual vs. Predicted Effort using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for New Web Projects



(a) (b) (c)

Figure 5.19: Actual vs. Predicted Effort using RF Technique based on Dataset 1, Dataset 2 and Dataset 3 for Enhanced Web Projects

Figures 5.18a, 5.18b and 5.18c display the final effort estimation model obtained using RF technique on new dataset 1, new dataset 2 and new dataset 3 respectively. Similarly, Figures 5.19a, 5.19b and 5.19c display the final effort estimation model obtained using RF technique on enhanced dataset 1, enhanced dataset 2 and enhanced dataset 3 respective;y. These figures show the variation of actual effort from the predicted result obtained using RF technique using these six categories of web projects. From these figure, it is observed that the difference between the actual and predicted value is on a higher side for all categories of web projects using RF technique than those obtained using DT and SGB techniques. These results can be easily validated

by finding out their corresponding error and prediction accuracy values provided in Tables 5.3 and 5.4.

5.4.4 Model Design using Various SVR Kernel Methods

After partitioning data into learning set and validation set, the model selection for ϵ and γ is performed using 10-fold cross validation process for the 3 categories of dataset of new and enhanced web projects. In this study, to perform model selection, the ϵ and γ values are varied over a range. The γ value ranges from 2^{-7} to 2^{7} and ϵ value ranges from 0 to 5. Hence, ninety number of models are generated to perform model selection operation.

The proposed model generated using the SVR linear, polynomial, RBF and sigmoid kernel for all six categories web datasets have been plotted below. These figures display the actual effort and the predicted effort obtained for web using the four SVR kernel methods taking into consideration six different project datasets.

New Dataset 1

SVR Linear Kernel Result:

Param: -s 3 -t 0 -c 0.99997 -g 0.0078125 -p 0

* Mean Squared Error (MSE_TEST) = 0.0030

* Squared correlation coefficient = 0.7634

SVR Polynomial Kernel Result:

Param: -s 3 -t 1 -c 0.99997 -g 0.5 -p 0

* Mean Squared Error (MSE_TEST) = 0.0053

* Squared correlation coefficient = 0.5952

SVR RBF Kernel Result:

Param: -s 3 -t 2 -c 0.99997 -g 0.5 -p 0

* Mean Squared Error (MSE_TEST) = 0.0031

* Squared correlation coefficient = 0.7630

SVR Sigmoid Kernel Result:

Param: -s 3 -t 3 -c 0.99997 -g 0.5 -p 0

* Mean Squared Error (MSE_TEST) = 0.0046

* Squared correlation coefficient = 0.7657

New Dataset 2

SVR Linear Kernel Result:

Param: -s 3 -t 0 -c 0.99961 -g 0.0078125 -p

* Mean Squared Error (MSE_TEST) = 0.0038

* Squared correlation coefficient = 0.9159

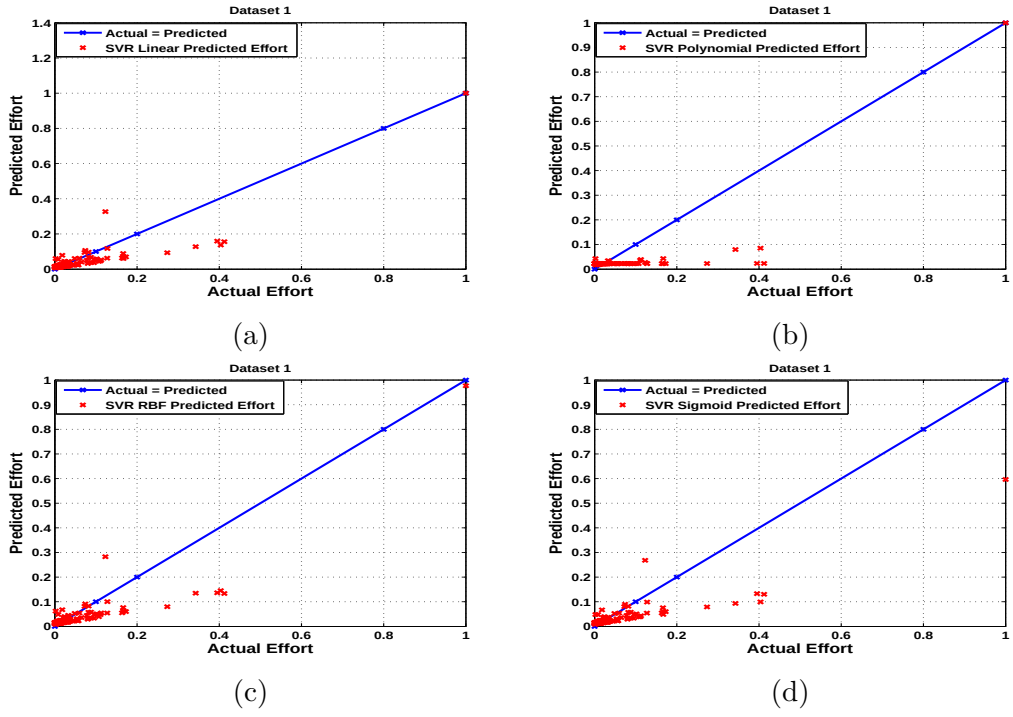


Figure 5.20: SVR Linear, Polynomial, RBF and Sigmoid Kernel based Web Software Effort Estimation using New Dataset 1

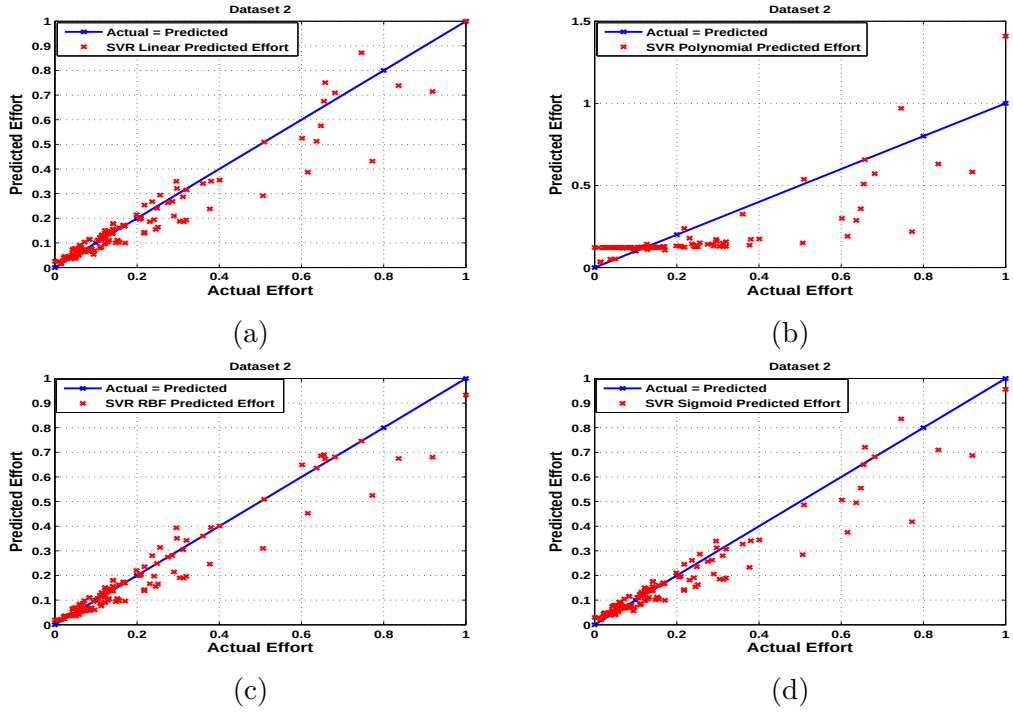


Figure 5.21: SVR Linear, Polynomial, RBF and Sigmoid Kernel based Web Software Effort Estimation using New Dataset 2

SVR Polynomial Kernel Result:

Param: -s 3 -t 1 -c 0.99961 -g 64 -p 0

* Mean Squared Error (MSE_TEST) = 0.0165

* Squared correlation coefficient = 0.6244

SVR RBF Kernel Result:

Param: -s 3 -t 2 -c 0.99961 -g 16 -p 0

* Mean Squared Error (MSE_TEST) = 0.0031

* Squared correlation coefficient = 0.9334

SVR Sigmoid Kernel Result:

Param: -s 3 -t 3 -c 0.99961 -g 0.25 -p 0

* Mean Squared Error (MSE_TEST) = 0.0042

* Squared correlation coefficient = 0.9164

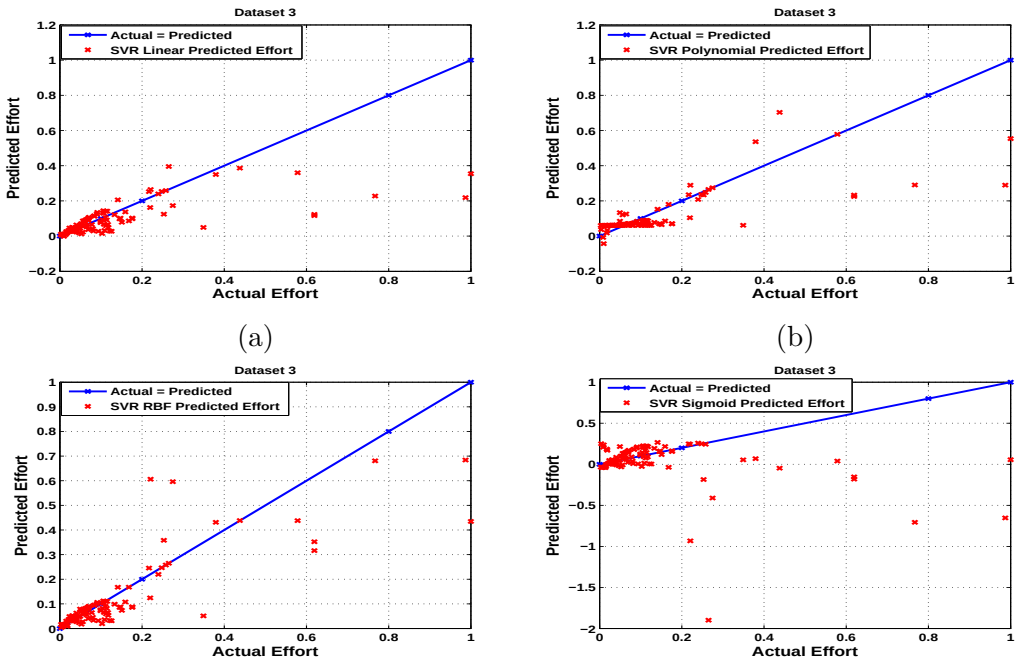


Figure 5.22: SVR Linear, Polynomial, RBF and Sigmoid Kernel based Web Software Effort Estimation using New Dataset 3

New Dataset 3

SVR Linear Kernel Result:

Param: -s 3 -t 0 -c 0.99683 -g 0.0078125 -p 0

* Mean Squared Error (MSE_TEST) = 0.0202

* Squared correlation coefficient = 0.4720

SVR Polynomial Kernel Result:

Param: -s 3 -t 1 -c 0.99683 -g 2 -p 0

* Mean Squared Error (MSE_TEST) = 0.0149

* Squared correlation coefficient = 0.5712

SVR RBF Kernel Result:

Param: -s 3 -t 2 -c 0.99683 -g 8 -p 0

* Mean Squared Error (MSE_TEST) = 0.0102

* Squared correlation coefficient = 0.7002

SVR Sigmoid Kernel Result:

Param: -s 3 -t 3 -c 0.99683 -g 1 -p 0

* Mean Squared Error (MSE_TEST) = 0.1439

* Squared correlation coefficient = 0.1235

The proposed model generated values for new project dataset 1, 2 and 3 using the SVR linear, polynomial, RBF and sigmoid kernel have been plotted as shown in Figures 5.20a, 5.20b, 5.20c and 5.20d, 5.21a, 5.21b, 5.21c and 5.21d, 5.22a, 5.22b, 5.22c and 5.22d respectively. These figures display the variation of actual effort and the predicted effort obtained using the four SVR kernel methods taking into consideration 3 categories of new web project dataset.

Enhanced Dataset 1

SVR Linear Kernel Result:

Param: -s 3 -t 0 -c 0.99987 -g 0.0078125 -p 0

* Mean Squared Error (MSE_TEST) = 0.0149

* Squared correlation coefficient = 0.2321

SVR Polynomial Kernel Result:

Param: -s 3 -t 1 -c 0.99987 -g 1 -p 0

* Mean Squared Error (MSE_TEST) = 0.0150

* Squared correlation coefficient = 0.1723

SVR RBF Kernel Result:

Param: -s 3 -t 2 -c 0.99987 -g 2 -p 0

* Mean Squared Error (MSE_TEST) = 0.0092

* Squared correlation coefficient = 0.5133

SVR Sigmoid Kernel Result:

Param: -s 3 -t 3 -c 0.99987 -g 1 -p 0

* Mean Squared Error (MSE_TEST) = 0.0163

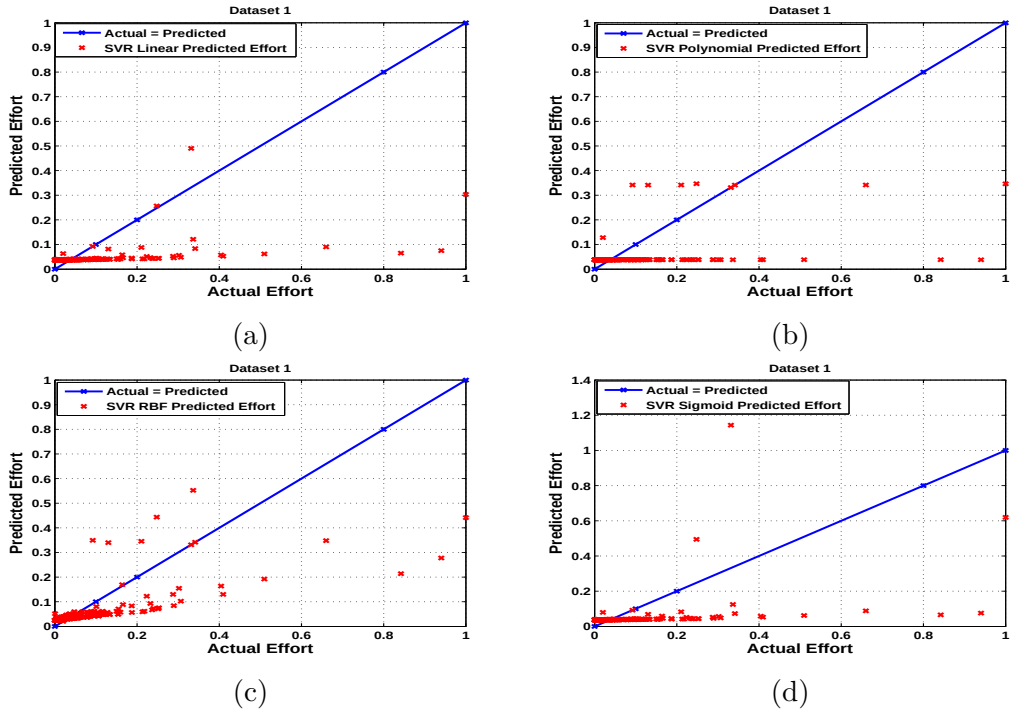


Figure 5.23: SVR Linear, Polynomial, RBF and Sigmoid Kernel based Web Software Effort Estimation using Enhanced Dataset 1

* Squared correlation coefficient = 0.1538

Enhanced Dataset 2

SVR Linear Kernel Result:

Param: -s 3 -t 0 -c 0.99931 -g 0.0078125 -p 0

* Mean Squared Error (MSE_TEST) = 0.0028

* Squared correlation coefficient = 0.9284

SVR Polynomial Kernel Result:

Param: -s 3 -t 1 -c 0.99931 -g 1 -p 0

* Mean Squared Error (MSE_TEST) = 0.0064

* Squared correlation coefficient = 0.6509

SVR RBF Kernel Result:

Param: -s 3 -t 2 -c 0.99931 -g 0.0625 -p 0

* Mean Squared Error (MSE_TEST) = 0.0012

* Squared correlation coefficient = 0.9316

SVR Sigmoid Kernel Result:

Param: -s 3 -t 3 -c 0.99931 -g 0.25 -p 0

* Mean Squared Error (MSE_TEST) = 0.0015

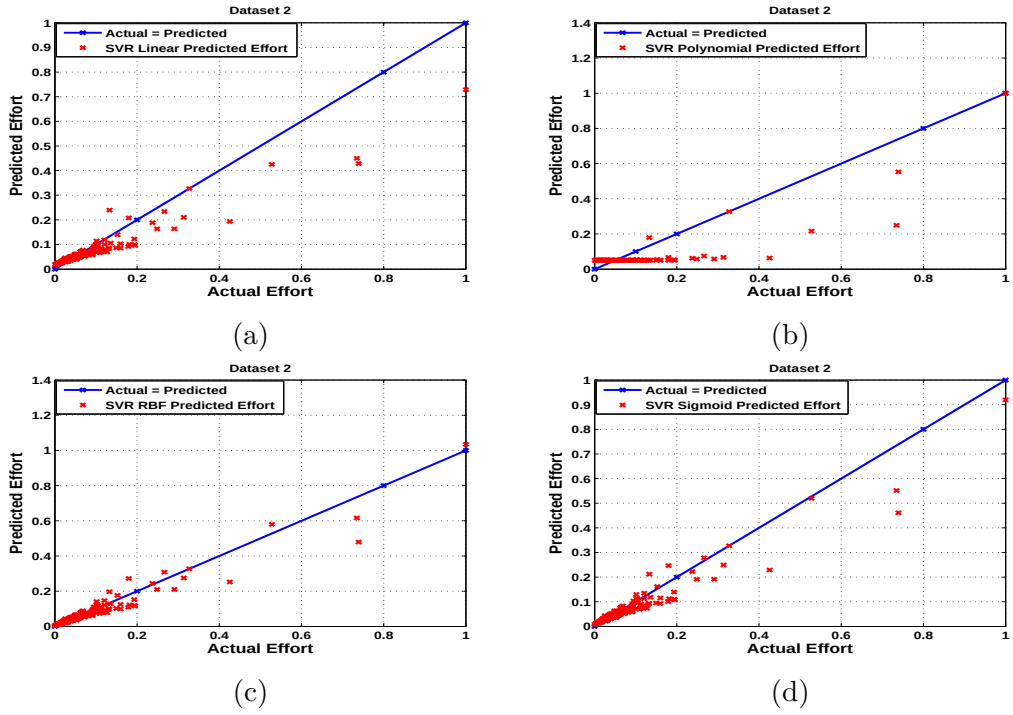


Figure 5.24: SVR Linear, Polynomial, RBF and Sigmoid Kernel based Web Software Effort Estimation using Enhanced Dataset 2

* Squared correlation coefficient = 0.9339

Enhanced Dataset 3

SVR Linear Kernel Result:

Param: -s 3 -t 0 -c 0.45438 -g 0.0078125 -p 0

* Mean Squared Error (MSE_TEST) = 0.0036

* Squared correlation coefficient = 0.8669

SVR Polynomial Kernel Result:

Param: -s 3 -t 1 -c 0.45438 -g 2 -p 0

* Mean Squared Error (MSE_TEST) = 0.0111

* Squared correlation coefficient = 0.4800

SVR RBF Kernel Result:

Param: -s 3 -t 2 -c 0.45438 -g 2 -p 0

* Mean Squared Error (MSE_TEST) = 0.0056

* Squared correlation coefficient = 0.7560

SVR Sigmoid Kernel Result:

Param: -s 3 -t 3 -c 0.45438 -g 2 -p 0

* Mean Squared Error (MSE_TEST) = 0.0027

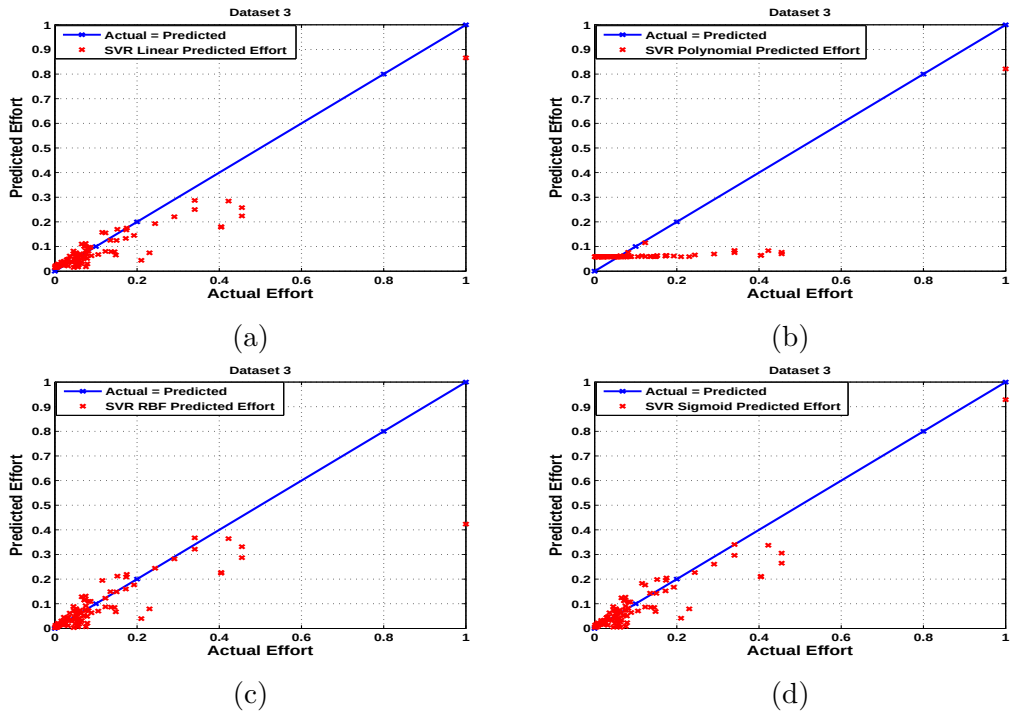


Figure 5.25: SVR Linear, Polynomial, RBF and Sigmoid Kernel based Web Software Effort Estimation using Enhanced Dataset 3

* Squared correlation coefficient = 0.8757

Similarly, the proposed model generated values for enhancement project 1, 2 and 3 using the SVR linear, polynomial, RBF and sigmoid kernel have been as shown in Figures 5.23a, 5.23b, 5.23c and 5.23d, 5.24a, 5.24b, 5.24c and 5.24d, 5.25a, 5.25b, 5.25c and 5.25d respectively. These figures display the variation of actual effort and the predicted effort obtained using the various SVR kernel methods technique taking into consideration 3 types of enhancement web project dataset. In these graphs, it may be observed that the data points are very little dispersed than the regression line. Hence the correlation is higher especially in case of enhancement project dataset 2. While comparing the dispersion of data points from the predicted model in the above graphs, it is observed that in case of both the new and enhanced web projects, the data points are less dispersed for SVR RBF kernel based model than other models. Hence, this model exhibits less error values and higher prediction accuracy value.

5.5 Comparison & Analysis of Result

The estimated effort value using DT, SGB, RF and our SVR kernel techniques are compared based on the results obtained. The DT uses recursive partitioning method

to split each node is split into two nodes with the help of a splitting variable. The SGB creates a tree ensemble, and also uses randomization during the creations of the trees. The prediction accuracy is calculated by feeding the result obtained from one tree to

Table 5.2: Comparison of MMRE, MdMRE and Prediction Accuracy Values of Related Works

Article	Dataset	Techniques		MMRE	MdMRE	PRED (25)
Mendes [77]	150 Projects from Tukutuku Dataset	Baysian Network (BN)		0.343	0.274	33.3
		Forward Step Wise Regression (FSWR)		0.948	1.00	6.7
		Case Based Reasoning (CBR)	One Analogy (CBR1)	1.381	0.851	13.3
			Two Analogy (CBR2)	1.347	0.85	13.3
			Three Analogy (CBR3)	2.03	0.917	13.3
		Classification and Regression Tree (CART)		6.904	0.133	20
Mendes and Mosley [78]	130 Projects from Tukutuku Dataset Divided into two separate dataset each containing 65 Projects	Manual Step Wise Regression (MSWR)	Validation Set 1	1.50	0.64	23.08
			Validation Set 2	0.73	0.66	16.77
Corazza et al. [13]	Tukutuku Dataset	Tabu Search + Support Vector Regression (TS+SVR)	No Transformation Linear Kernel (C1(Lin))	1.989	0.582	23.4
			No Transformation RBF Kernel (C1(RBF))	1.712	0.663	29.7
			Logarithimic Transformation Linear Kernel (C3(Lin))	1.151	0.552	28.1
			Logarithimic Transformation RBF Kernel (C3(RBF))	0.590	0.339	39.1
Corazza et al. [14]	Tukutuku Dataset	SVR with Logarithimic Transformation and RBF Kernel (SVR with C3(R))	First Set	0.59	0.41	34
			Second Set	0.91	0.36	42
			Third Set	1.10	0.33	42
Ferrucci et al. [82]	195 Projects from Tukutuku Dataset	Manual Step Wise Regression + Linear Regression for Cross Company Dataset	Validation Set 1 (S5CCM1)	0.81	0.61	6
			Validation Set 2 (S5CCM2)	0.58	0.61	13
		Manual Step Wise Regression + Linear Regression for Single Company Dataset	Validation Set 1 (S5SCM1)	0.33	0.23	52
			Validation Set 2 (S5SCM2)	0.37	0.27	50

the next tree in the series. However, RF builds trees in parallel and also uses voting method on the prediction. The RF uses the "out of bag" data rows for validation. This aspect provides an independent test without requiring a separate data set or holding back rows from the tree construction. Table 5.2 gives a relative investigation of the outcomes acquired by a few articles specified in the related work section. The

MMRE, MdMRE and prediction accuracy (PRED(25)) values are taken as a measure in order to evaluate the performance obtained using techniques mentioned in those articles. Results indicate that, a maximum of 52% prediction accuracy is achieved using the Hybrid technique (Manual Step Wise Regression and Linear Regression) for web-based applications. At last, the outcomes acquired from the techniques presented in related work section is measured against the proposed methodology as displayed in Tables 5.3 and 5.4. The results obtained using proposed techniques show enhancement in prediction accuracy value.

Table 5.3: Comparison of Results of Three Categories of Dataset using DT, SGB, RF and four SVR Kernels for New Web Projects

	Dataset Category	RMSE	MAE	MMRE	MMER	PRED(25)	PRED(50)	PRED(75)	PRED(100)
DT	**Dataset 1**	0.0613	0.0248	6.3851	0.4998	26.4286	60	74.2857	80
	Dataset 2	0.0450	0.0303	0.8133	0.2048	59.3548	93.5484	96.7742	97.5806
	Dataset 3	0.1347	0.0806	1.8570	0.6373	25.9615	47.1154	59.6154	70.1923
SGB	**Dataset 1**	0.0604	0.0207	5.2902	0.4032	38.5714	68.5714	76.4286	80
	Dataset 2	0.0448	0.0257	0.6649	0.1689	76.6129	92.7419	97.5806	98.3871
	Dataset 3	0.1057	0.0545	0.7949	0.4368	28.8462	63.4615	75.9615	80.7692
RF	**Dataset 1**	0.0918	0.0317	8.4046	0.6068	27.1429	57.1429	68.5714	75.7143
	Dataset 2	0.0891	0.0552	2.4085	0.2945	50.8065	74.1935	80.6452	85.4839
	Dataset 3	0.1151	0.0589	1.0307	0.4954	28.8462	60.5769	75	81.7308
SVR Linear Kernel	**Dataset 1**	0.0555	0.0262	6.0081	0.5929	38.5714	58.5714	80	82.1429
	Dataset 2	0.0620	0.0363	0.7416	0.2330	65.3226	91.9355	96.7742	98.3871
	Dataset 3	0.1421	0.0609	0.3940	0.6365	34.6154	73.0769	89.4231	98.0769
SVR Polynomial Kernel	**Dataset 1**	0.0729	0.0349	9.3727	1.3566	23.5714	40.7143	61.4286	78.5714
	Dataset 2	0.1284	0.0843	3.2419	0.5171	37.9032	63.7097	78.2258	81.4516
	Dataset 3	0.1223	0.0592	1.0475	0.5001	36.5385	57.6923	75.9615	79.8077
SVR RBF Kernel	**Dataset 1**	0.0545	0.0257	5.6497	0.5584	36.4286	57.1429	80	83.5714
	Dataset 2	0.0556	0.0323	0.5897	0.2255	68.8710	96.7742	98.3871	99.1935
	Dataset 3	0.1011	0.0479	0.3020	0.4485	39.4231	72.1154	91.3462	94.2308
SVR Sigmoid Kernel	**Dataset 1**	0.0678	0.0296	5.8261	0.6394	36.4286	57.8571	78.5714	83.5714
	Dataset 2	0.0647	0.0377	0.8576	0.2400	62.9032	89.5161	95.9677	97.5806
	Dataset 3	0.3794	0.1657	2.3898	1.2978	21.1538	31.7308	36.5385	56.7308

Tables 5.3 and 5.4 show the performance of various machine learning techniques in effort estimation of web projects. These three techniques are applied separately over six types of web projects for generating their corresponding effort models.

Figures 5.26a, 5.26b, 5.26c, 5.26d, 5.26e and 5.26f display the box plot using Dataset 1, Dataset 2 and Dataset 3 for new web project respectively to illustrate the spread and differences of samples, with the help of their corresponding Error and

Table 5.4: Comparison of Results of Three Categories of Dataset using DT, SGB, RF and four SVR Kernels for Enhanced Web Projects

	Dataset Category	RMSE	MAE	MMRE	MMER	PRED (25)	PRED (50)	PRED (75)	PRED (100)
DT	**Dataset 1**	0.0626	0.0293	1.3448	0.4472	32.3887	62.7530	72.8745	77.3279
	Dataset 2	0.0483	0.0200	0.3957	0.2509	59.5092	88.9571	93.2515	93.8650
	Dataset 3	0.0643	0.0307	1.1192	0.4295	47.5248	63.3663	78.2178	82.1782
SGB	**Dataset 1**	0.0701	0.0295	1.6863	0.4190	32.3887	62.3482	67.6113	73.2794
	Dataset 2	0.0347	0.0181	0.3219	0.2495	60.1227	89.5706	93.2515	95.7055
	Dataset 3	0.0687	0.0295	1.1425	0.3780	44.5545	67.3267	76.2376	82.1782
RF	**Dataset 1**	0.0862	0.0367	3.2330	0.5278	26.7206	52.2267	60.7287	66.8016
	Dataset 2	0.0701	0.0357	1.6007	0.4230	38.6503	65.0307	73.6196	75.4601
	Dataset 3	0.0938	0.0439	2.1201	0.5083	38.6139	59.4059	71.2871	76.2376
SVR Linear Kernel	**Dataset 1**	0.1220	0.0580	4.0857	1.1747	17.4089	30.7692	55.0607	65.9919
	Dataset 2	0.0529	0.0262	0.7677	0.3466	47.2393	74.8466	81.5951	88.3436
	Dataset 3	0.0596	0.0348	0.9449	0.5627	33.6634	57.4257	77.2277	83.1683
SVR Polynomial Kernel	**Dataset 1**	0.1223	0.0598	4.2886	1.3961	17.4089	30.3644	51.4170	65.1822
	Dataset 2	0.0801	0.0483	1.9657	0.8145	18.4049	42.3313	69.3252	76.0736
	Dataset 3	0.1052	0.0615	2.8180	0.9265	33.6634	43.5644	60.3960	73.2673
SVR RBF Kernel	**Dataset 1**	0.0959	0.0468	2.9739	0.7385	16.1943	40.8907	65.5870	70.4453
	Dataset 2	0.0262	0.0137	0.2763	0.1890	69.3252	92.0245	96.9325	96.9325
	Dataset 3	0.0751	0.0348	0.5130	0.3560	30.6931	66.3366	85.1485	93.0693
SVR Sigmoid Kernel	**Dataset 1**	0.1278	0.0605	4.0995	1.1741	17.0040	30.3644	54.2510	65.5870
	Dataset 2	0.0394	0.0199	0.4325	0.2806	56.4417	85.2761	91.4110	92.6380
	Dataset 3	0.0519	0.0315	0.5789	0.3583	30.6931	64.3564	82.1782	93.0693

MER values generated using DT, SGB, RF and four SVR kernel techniques. Similarly, Figures 5.27a, 5.27b, 5.27c, 5.27d, 5.27e and 5.27f display the box plot using Dataset 1, Dataset 2 and Dataset 3 for enhanced web project respectively to illustrate the spread and differences of samples with the help of their corresponding Error and MER values generated using DT, SGB, RF and four SVR kernel techniques respectively. By analyzing the results given in the above tables and figures, it is observed that for both new (dataset 1, 2 and 3) and enhanced (dataset 1, 2 and 3) web projects, SVR RBF technique outperforms other techniques for effort estimation purpose.

In order to affirm the robustness of the proposed models, the non-parametric Mann-Whitney p-value test and effect size [131] tests such as Cohen's d and Glass's Δ between diverse proposed models are processed considering absolute residuals as demonstrated in Tables 5.5 and 5.6. Results demonstrate that for both new and enhanced web projects, all the models are statistically significant at the 95% confidence internal i.e., p-value value < 0.05. From the results provided in Tables 5.5 and 5.6, it is evident that the effect size is mostly small in every cases for new as well

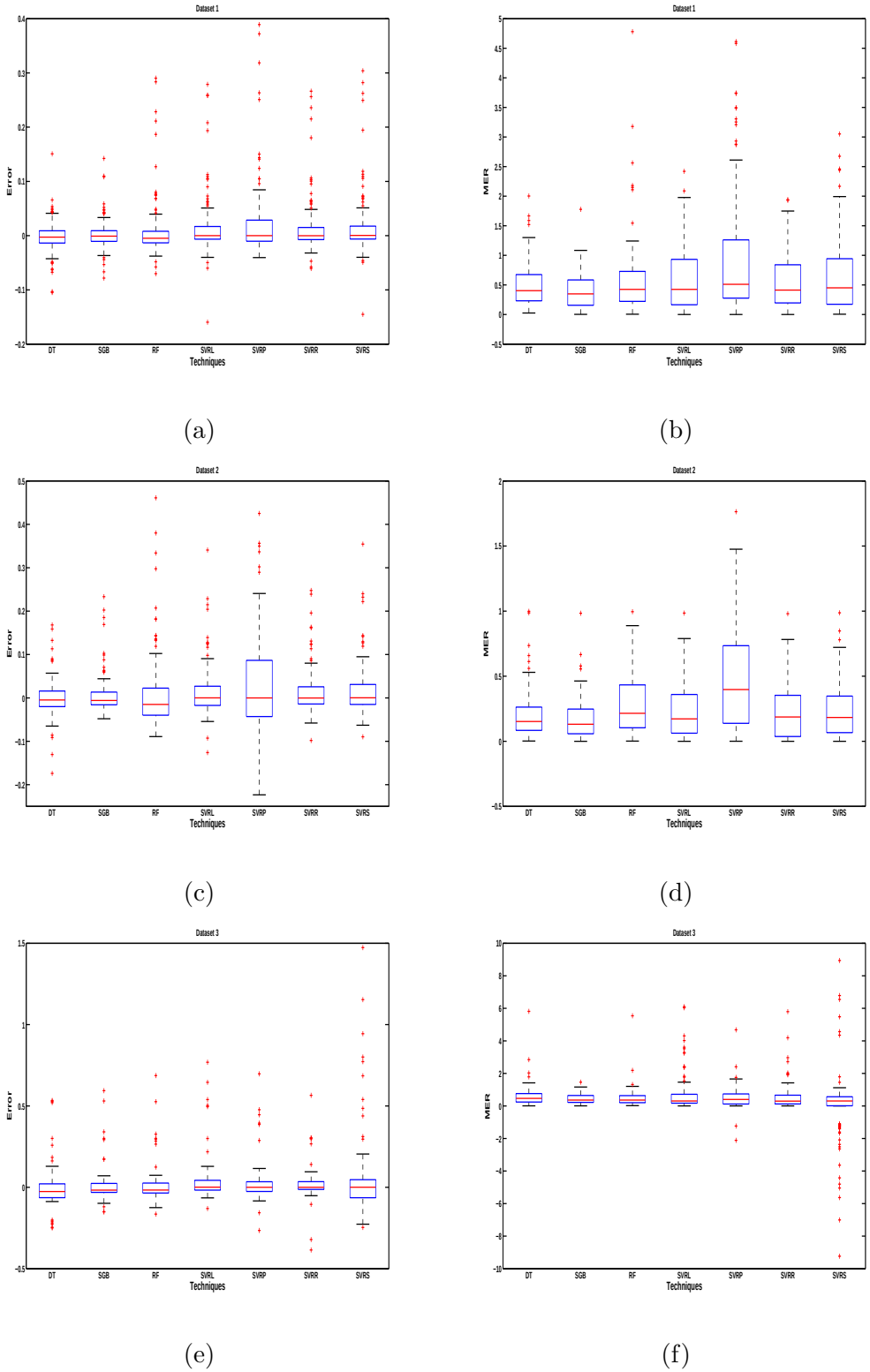


Figure 5.26: Boxplots of Errors and MERs for Dataset 1, 2 and 3 of New Web Projects as enhanced web projects.

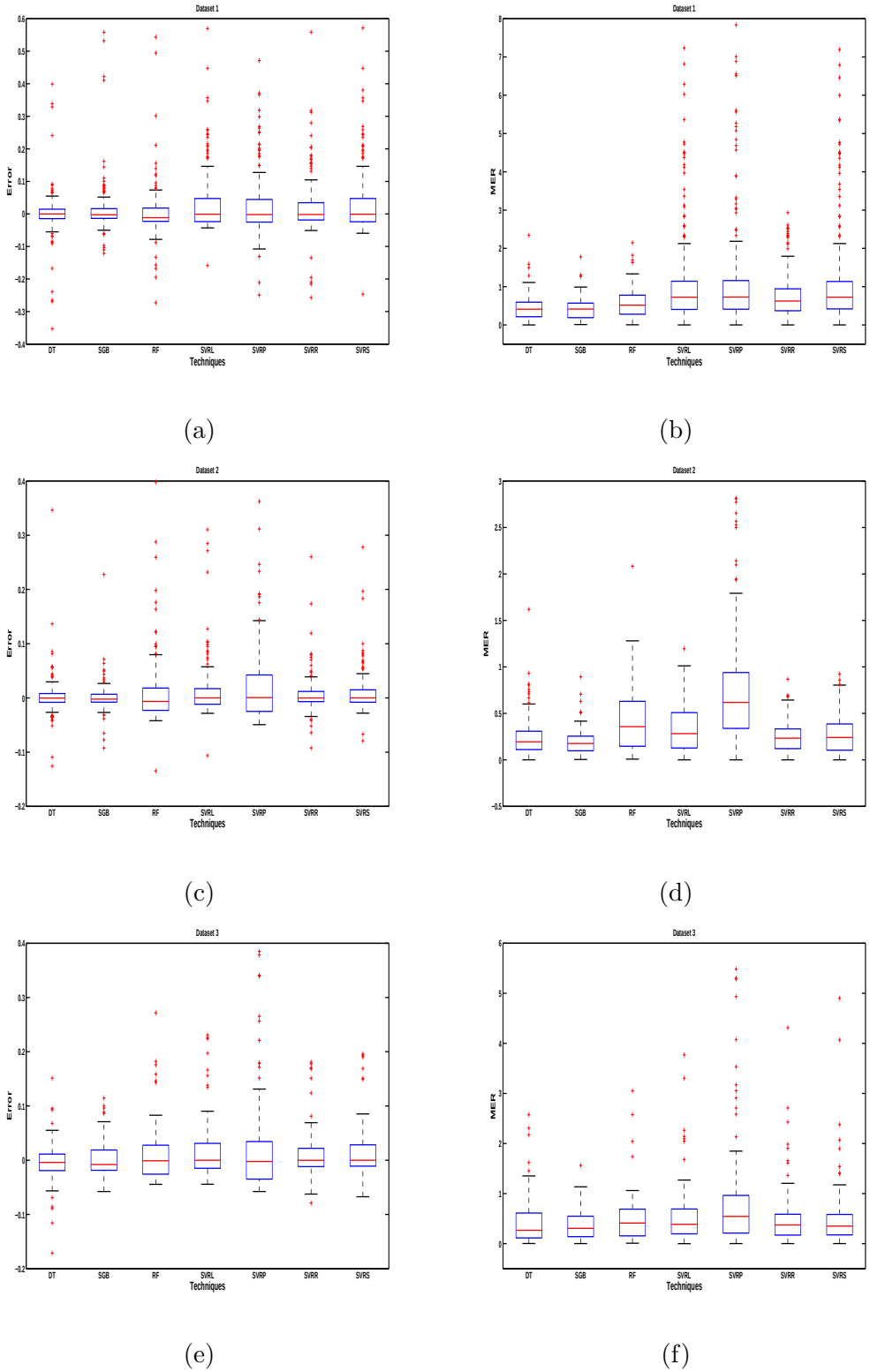


Figure 5.27: Boxplots of Errors and MERs for Dataset 1, 2 and 3 of Enhanced Web Projects

Table 5.5: Comparison of Statistical Significance and Effect Size Test of Proposed Models for New Web Projects

		Effect Size Test	
	Mann-Whitney p-Value	**Cohen's d**	**Glass's Δ**
DT1 vs. SVR RBF1	0.02642	0.1421	0.1418
SGB1 vs. SVR RBF1	0.03352	0.0751	0.0654
RF1 vs. SVR RBF1	0.00634	0.0348	0.0260
DT2 vs. SVR RBF2	0.04364	0.0768	0.0794
SGB2 vs. SVR RBF2	0.04706	0.0447	0.0443
RF2 vs. SVR RBF2	0.00072	0.0190	0.0162
DT3 vs. SVR RBF3	< 0.00001	0.1529	0.1409
SGB3 vs. SVR RBF3	0.00008	0.0747	0.0660
RF3 vs. SVR RBF3	0.00018	0.0458	0.0392

Table 5.6: Comparison of Statistical Significance and Effect Size Test of Proposed Models for Enhanced Web Projects

		Effect Size Test	
	Mann-Whitney p-Value	**Cohen's d**	**Glass's Δ**
DT1 vs. SVR RBF1	0.04352	0.2342	0.3095
SGB1 vs. SVR RBF1	0.03788	0.1619	0.1722
RF1 vs. SVR RBF1	0.00094	0.2036	0.2181
DT2 vs. SVR RBF2	0.03953	0.0537	0.0542
SGB2 vs. SVR RBF2	0.01770	0.0490	0.0495
RF2 vs. SVR RBF2	0.01778	0.0526	0.0428
DT3 vs. SVR RBF3	0.03752	0.1568	0.1842
SGB3 vs. SVR RBF3	0.04012	0.0902	0.0907
RF3 vs. SVR RBF3	0.03953	0.0320	0.0271

5.6 Summary

In this chapter, the ISBSG dataset has been used for developing effort estimation models for web-based projects (new and enhanced) using IFPUG Function Point approach. Different machine learning techniques such as Decision Tree, Stochastic Gradient Boosting, Random Forest and four SVR kernel techniques are employed on ISBSG dataset. From the analysis of the result, it is evident that in case of both the new and enhanced web projects, SVR RBF kernel technique exhibits better results than other ML techniques, as it provides minimal error and higher prediction accuracy on the six types of considered project datasets. The computations for above methodologies were executed, and results were obtained using MATLAB.

Chapter 6

Story Point Approach for Agile Software Effort Estimation using Machine Learning Techniques

6.1 Introduction

Agile methods are used for developing software to enable organizations respond to requirements volatility. These methods provide opportunities to assess the direction of software development throughout the development life cycle [149]. By emphasizing on the repetition of work cycles along with product the teams it leads to an additive and iterative development. Instead of promising to market an ensemble software that hasn't been developed, agile authorizes teams to repetitively re-plan their releases in order to optimize their value throughout the development [96, 109]. Thus, companies following agile methods bring competition to others in the marketplace that don't use agile methods [150].

Since predictability of requisite resources is the primary goal at the starting phase of project management, to estimate the size and complexity of the products to be built in order to determine what to do next becomes the focus in agile development process [151, 152]. For this purpose of prediction of resources, requirements need to be collected. Requirements in agile development are jotted down in cards and are called user stories [153, 154]. These stories are estimated using story points. The team defines the relationship between story point and effort. Usually 1 story point is equal to 1 ideal working day. Total no. of story points that a team can convey in a sprint (an iteration in agile software development) is called as "team velocity" or story points per sprint. In the story point approach, total number of story points are used along with project velocity to determine the effort required for agile software development. Now for obtaining better prediction accuracy, Random Forest and four SVR kernel techniques are applied on the story point dataset. The results obtained

by applying these machine learning techniques are compared among themselves as well as with the results obtained by other author available in the literature and their performance is assessed.

6.2 Methodology Used

The methodology described below are utilized as a part of this chapter in order to ascertain the effort of a software developed using agile methodology.

6.2.1 Story Point Approach (SPA)

Story Point is a unit to quantify the size of a user story or feature. A story point might be assigned in view of the effort included, the complexity and the inalienable risk in building up a story [95, 155]. An appraisal of the effort of building up a user story requires the designer to have some experience of evaluating, to have admittance to historical data and have the opportunity to utilize a trial-based estimation approach. The block diagram, appeared in Figure 6.1, demonstrates the steps to ascertain the project development effort utilizing story point approach [156, 157].

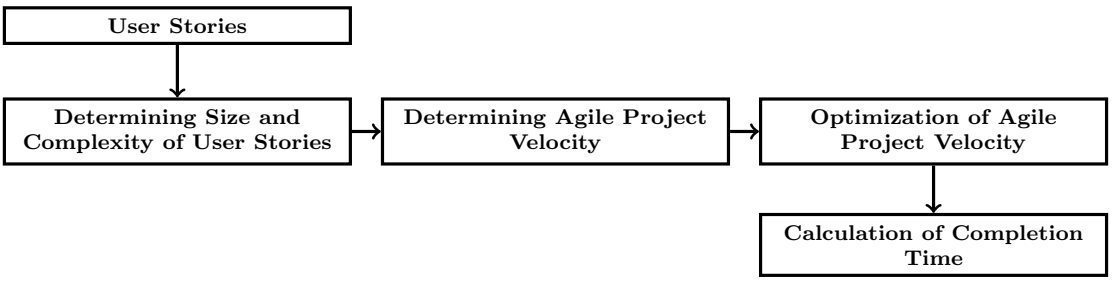


Figure 6.1: Steps to Calculate Effort Using Story Point Approach

After developing user stories, the size and the complexity of each of the user stories need to be found out for calculating story point. Story size is an estimate of the relative scale of the work in terms of actual development effort. It can be calculated in a scale of 1 to 5 where 1 means a very small story representing tiny effort level and 5 means an extremely large story [97].

Story complexity indicates either the requirements complexity or its technical complexity. Complexity introduces uncertainty to the estimate, i.e., more complexity means more uncertainty. Similar to story size calculation, the complexity of each user story can also be determined in a scale of 1 to 5. In this case, complexity level 1 means a very less complex story, where technical and business requirements are very clear with less amount of ambiguity. It requires basic programming skills to complete. Similarly, complexity level 5 means extremely complex story which has

a many dependencies on other stories or other systems or subsystems. It requires significant refactoring, extensive research and difficult judgment calls to complete.

Using these two vectors, effort of a particular user story (ES) is determined as:

$$ES = Complexity \times Size \tag{6.1}$$

Effort for the complete project will be sum of efforts of all individual user stories. It is calculated as:

$$Effort(E) = \sum_{i=1}^{n} (ES)_i \tag{6.2}$$

The unit of $Effort(E)$ is Story Point (SP).

In Agile term, velocity can be defined as how much product backlog effort a team can handle in one unit of time. The initial / raw velocity (Vi) is calculated as:

$$V_i = \frac{Units\ of\ Effort\ Completed}{Sprint\ Time} \tag{6.3}$$

This observed velocity describes how many units of effort team completes in a typical Sprint. During the development of a product, a sprint is typically a time frame amid which particular work need to be finished and made prepared for review. The scrum Master finalizes a teams sprint length. The sprint length varies from organization to organization as well as from product to product in a single organization.

Calibration can start, only when the process of optimization is finished [97]. There are two different aspects on calibration. They are:

- The Friction or consistent forces drag continually on productivity and minimizes project velocity.
- The Variable or Dynamic Forces decelerate the project or team members and cause the project velocity to be sporadic.

Optimizing both of these components before calibration will enhance the dependability in evaluating the project velocity.

Table 6.1: Friction Factors [97]

Sl. No.	**Friction Factors**	**Stable**	**Volatile**	**Highly Volatile**	**Very Highly Volatile**
1	Team Composition	1	0.98	0.95	0.91
2	Process	1	0.98	0.94	0.89
3	Environmental Factors	1	0.99	0.98	0.96
4	Team Dynamics	1	0.98	0.91	0.85

Friction (FR) is calculated as product of all four fraction factors (FF) provided in Table 6.1.

$$FR = \prod_{i=1}^{4}(FF)_i \tag{6.4}$$

Variable or Dynamic forces are frequently eccentric and unforeseen. They decelerate the project and also cause a loss in project velocity.

Table 6.2: Dynamic Forces [97]

Sl. No.	**Variable Factors**	**Normal**	**High**	**Very High**	**Extremely High**
1	Expected Team Changes	1	0.98	0.95	0.91
2	Introduction of New Tools	1	0.98	0.94	0.89
3	Vendor's Defect	1	0.99	0.98	0.96
4	Team members responsibilities outside the project	1	0.98	0.91	0.85
1	Personal Issues	1	0.98	0.95	0.91
2	Expected Delay in Stakeholder response	1	0.98	0.94	0.89
3	Expected Ambiguity in Details	1	0.99	0.98	0.96
4	Expected Changes in environment	1	0.98	0.91	0.85
4	Expected Relocation	1	0.98	0.91	0.85

Dynamic Force (DF) is calculated as product of all nine variable factors (VF).

$$DF = \prod_{i=1}^{9}(VF)_i \tag{6.5}$$

Deceleration (D) is the product of Friction and Dynamic Forces affecting the velocity. It is calculated as:

$$D = FR \times DF \tag{6.6}$$

In order to adjust Velocity to more predictable range, the Final Velocity (V) is calculated as:

$$V = (V_i)^D \tag{6.7}$$

Hence, the estimated duration required to complete the project is calculated as:

$$T = \frac{\sum_{i=1}^{n}(ES)_i}{(Vi)^D} \times \frac{1}{WD} months \tag{6.8}$$

Where T denotes the completion time of the project and WD denotes the number of work days per month. In this study, the unit of T is calculated as months.

The total number of story points and the final velocity value of the agile projects

are then taken as input arguments to various machine learning models to calculate normalized effort.

6.3 Proposed Approach

The proposed approach is implemented using the twenty one project data set developed by six software houses [97]. In the data set, every row contains three columns. The first column indicates the number story points required to complete the project, the second column represents the velocity of the project, and the third column represents the actual effort required to complete that project. This data set is used to determine software development effort. Now in order to improve the effort predictions, various SVR kernel and RF techniques are applied. The statistical profile of dataset based on story point approach for agile software effort estimation is depicted in Table 6.3.

Table 6.3: Statistical Profile of Datasets based on Story Point Approach for Agile Software Effort Estimation

Project Type	Minimum	Maximum	Mean	Median	Std. Dev.	Skewness	Kurtosis
21 Project Dataset	21	112	56.43	52	26.18	0.65	-0.77

Figure 6.2 depicts the relationship between software size (total number of story point) and actual effort (person-hours) based on SPA using 21 project dataset. From

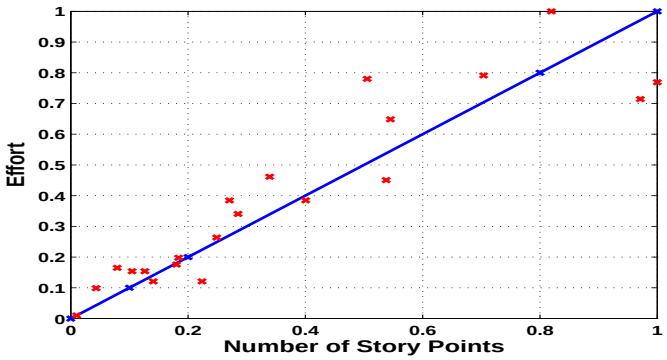


Figure 6.2: Software Size vs. Effort Graph based on Story Point Approach

these figures, it is observed that the 21 project dataset based on SPA contains few number of outliers. From Table 6.3, it has been observed that the dataset is not normally distributed based on the values of the skewness and kurtosis. Hence, in order to make the data normally distributed, logarithmic transformation is applied over the dataset.

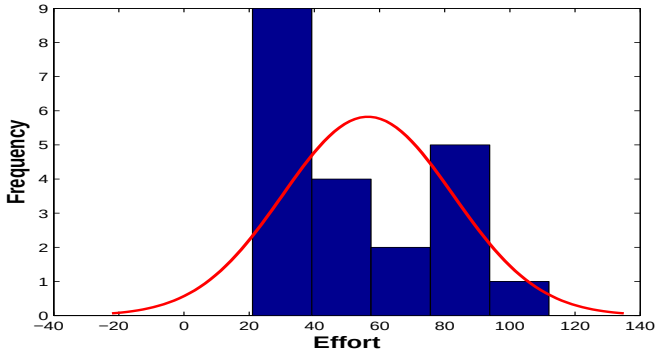


Figure 6.3: Histogram of Effort Values based on Story Point Approach

Figure 6.3 displays the histogram of effort value based on SPA for agile software effort estimation. From the figure, it can be observed that the data are more normally distributed. The block diagram, demonstrated in Figure 6.4, shows the proposed steps applied to predict effort with the help of RF and four SVR kernel techniques.

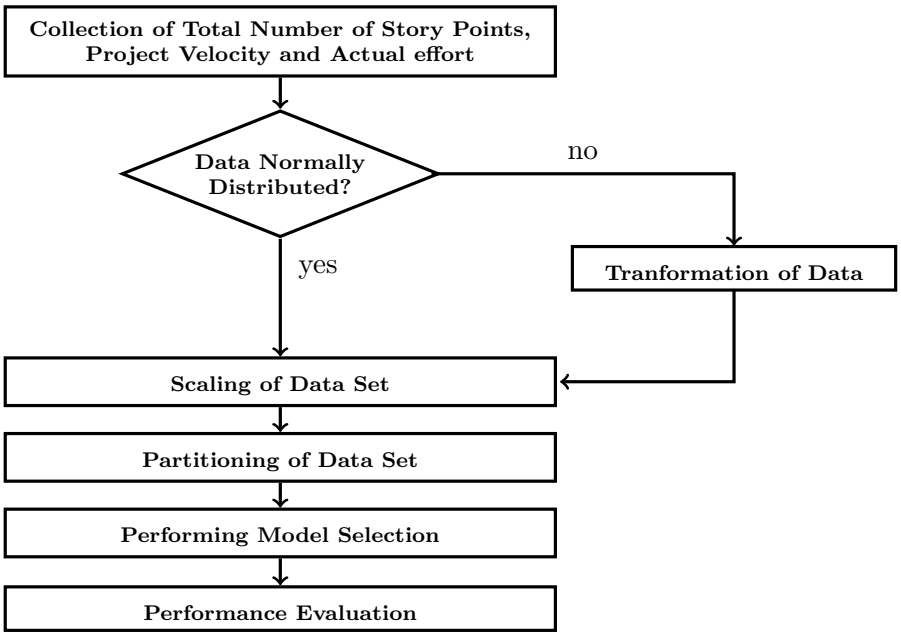


Figure 6.4: Proposed Steps for Software Effort Estimation Purpose applying RF and SVR Kernel Techniques

To compute the software development effort, essentially the accompanying steps are utilized.

Proposed Steps for Software Effort Estimation

1. **Collection of Total Number of Story Points, Project Velocity and Actual Effort** : The total number of story points, project velocity values and actual effort are collected from [97].

2. **Data Normally Distributed?**: The statistical analysis of the collected dataset has been performed and verified in order to check whether the collected dataset follows normal distribution or not based on the values of skewness and kurtosis. If data are normally distributed, then it will directly proceed to the data normalization step. Otherwise, the data need to be transformed to make it more normally distributed.

3. **Transformation of Data**: If the dataset is not normally distributed, then the logarithmic transformation has been applied over the dataset to make it normally distributed. Histograms have been plotted to properly verify the distribution of data before and after transformation.

4. **Scaling of Data Set** : In this step the input parameter values i.e., the total number of story points required to complete the project and the project velocity are individually scaled between the range 0 to 1. Let S represents the complete dataset and s represents a record in the S. Then the normalized value of s is obtained by using the following formula :

$$Normalized(s) = \frac{s - \min(S)}{\max(S) - \min(S)} \tag{6.9}$$

   where
   $\min(S)$ = min value in S.
   $\max(S)$ = max value in S.
   if $\min(S)$ is same as $\max(S)$, then Normalized(s) value is assigned as 0.5.

5. **Partitioning of dataset**: Total no. of data are divided into two subsets i.e., training set and test set for both RF and SVR Kernel techniques. Random forest have randomness in input data and in splitting at nodes. Hence, in case of RF technique, initially an arbitrary random vector is selected to provide randomness in input data and to start the implementation process. Then, the data are divided using this arbitrary random vector.

6. **Performing Model Selection**: In case of RF technique, prediction results vary according to random vector. So an evaluation function (1- MMER + Prediction Accuracy) is used to find a random vector. The random vector, which provides optimum value for the evaluation function is considered as final random vector. Then, by using this final random vector, results are being predicted.

   Similarly, in case of SVR kernel-based effort estimation model, the model which provides the least value in comparison with values of the other generated models based on the minimum validation error criteria has been selected to

perform other operations. The tunable parameters have been selected to find the best parameter C and γ using a five fold cross validation procedure. Based on the minimum validation error, the best model has been selected and the corresponding value of γ and ϵ value is found out. The final model selected based on best parameter of C, ϵ and γ has been trained using all training samples. The output of this step is the trained SVM model providing predicted response values for test inputs.

7. **Performance Evaluation**: In this study, the Mean Magnitude of Error Relative to the estimate (MMER) and the Prediction Accuracy (PRED(x)) are the two measures used to evaluate the performance of the model for test samples. Results obtained from proposed model-based on RF and SVR Kernel techniques are then evaluated against existing results to access its performance accuracy.

The ML techniques are implemented using the above steps. At last, a correlation of results acquired utilizing RF technique-based effort estimation model with other existing models is displayed in order to evaluate their performances.

6.4 Experimental Details

For implementing the proposed approaches, the data set given in [97] is used, provided in Table 6.4.

Out of these, initially total number of story points and project velocity are considered as input parameters to the machine learning model in order to assess their influences over predicted effort value. The total number of story point value is calculated by considering the size and complexity of individual stories. The project velocity is calculated by considering the various friction factors and dynamic factors values. The detailed description about the procedure to calculate these values are already been provided in section 6.2.1. Then, the input argument having the maximum influence is selected as the final input parameter to the model for calculating predicted effort. The utilization of such a dataset helps to calculate effort the required to develop a software using agile methodologies and provides introductory test information for the viability of the SPA. These data are utilized to obtain different machine learning technique-based effort estimation model. The output is the effort i.e., time required to complete the project.

Table 6.4: Twenty One Project Dataset based on SPA

#	Number of Story Points	Project Velocity	Actual Effort
1	**156**	**2.7**	**63**
2	**202**	**2.5**	**92**
3	**173**	**3.3**	**56**
4	**331**	**3.8**	**86**
5	**124**	**4.2**	**32**
6	**339**	**3.6**	**91**
7	**97**	**3.4**	**35**
8	**257**	**3**	**93**
9	**84**	**2.4**	**36**
10	**211**	**3.2**	**62**
11	**131**	**3.2**	**45**
12	**112**	**2.9**	**37**
13	**101**	**2.9**	**32**
14	**74**	**2.9**	**30**
15	**62**	**2.9**	**21**
16	**289**	**2.8**	**112**
17	**113**	**2.8**	**39**
18	**141**	**2.8**	**52**
19	**213**	**2.8**	**80**
20	**137**	**2.7**	**56**
21	**91**	**2.7**	**35**

6.4.1 Model Design Using Random Forest Technique

The Brieman's algorithm has been applied by a good number of authors to implement the random forest technique [19]. To design an effort estimation model using the random forest technique, the following steps are used. These proposed steps help in constructing each tree, while using random forest technique.

Steps of Proposed Algorithm:

1. Let F be the number of trees in the forest. A Dataset of D points $(x_1, y_1)(x_2, y_2)....(x_D, y_D)$ is considered.

2. Each tree of the forest should be grown as follows: Steps from i to vii should be repeated f times to create F number of trees.

   i. Let 'N' be the no. of training cases, and 'M' be the no. of variables in the classifier.

   ii. To select training set for the tree, a random sample of n cases - yet with substitution, from the original data of all 'N' accessible training cases is

chosen. Whatever is left of the cases are utilized to evaluate the error of the tree, by foreseeing their classes.

iii. A RF tree 'T_f' is developed to the loaded data, by repeatedly rehashing the accompanying steps for every terminal node of the tree, till the minimum node size 'n_{min}' is arrived. Keeping in mind the end goal to make more randomness, distinctive dataset for each one tree is made.

iv. The no. of input variables 'm' is selected to ascertain the choice at a tree node. The value of 'm' is assumed to be substantially short of what 'M'.

v. For each tree node, 'm' number of variables should be randomly chosen on which the decision at that node is based.

vi. The best split focused around these 'm' variables in the training set is calculated. The value of 'm' is assumed to be consistent throughout the development of the forest. Each tree should be fully grown and not pruned.

vii. Then, the results of ensemble of trees $T_1, T_2, ..., T_f, ..., T_F$ are collected.

3. The input vector should be put down for each of the trees in the forest. In regression, it is the average of prediction values of the individual tree predictions.

$$Y^F(x) = 1/F \sum_{f=1}^{F} T_f(x) \tag{6.10}$$

where

$Y^F(x)$ is the predicted value for the input vector x.

$T_1(x), T_2(x), ..., T_f(x)$ represents prediction value of individual trees.

There are various data objects generated by random forest technique, which needs to be considered while implementing random forest technique for software effort estimation purpose. The results obtained from these data objects need to be evaluated in order to assess the performance achieved using random forest technique.

Variable Importance

The variable importance defines the contribution of a variable in achieving accurate prediction. It is calculated by taking into consideration of its interaction with other variables. The error rate for each tree T, is calculated using the Out-of-Bag(OOB) data. Then, the permutation result of the OOB values is calculated for each variable 'v' and the error value is calculated again using each tree. If the number of variables for implementing RF technique is very large, forests can be run once with all the

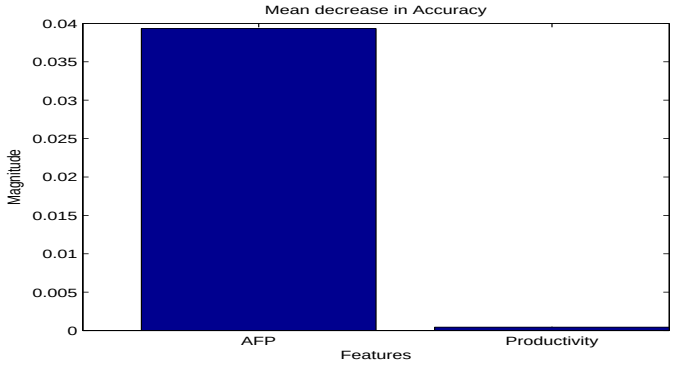


Figure 6.5: Variable Importance

variables. Then, by using only the most important variable from the initial run, forests can be run again to calculate the final predicted effort.

Figure 6.5 displays the importance of two variables i.e., total number of story points and project velocity taken as input to the model for calculating the effort using random forest technique. From the figure, it may be observed that the impact of the variable *story point* is the highest for predicting the effort required to develop the software. Therefore, total number of story point column is finally chosen for predicting effort in the proposed approach.

The Out-Of-Bag (OOB) Error Estimate

The training set for a tree is produced by testing with substitution. During this process, something like one-third of the cases are left out of the sample. These cases are considered as out-of-bag (OOB) data. It helps in getting an impartial evaluation of the regression error value as the forest develops. OOB data also helps in getting estimation of variable importance. In RF, as the value of OOB is calculated internally during the run, cross validation of data or a different test set to obtain an impartial evaluation of the test error is not required. The computation procedure for finding value of OOB is explained below.

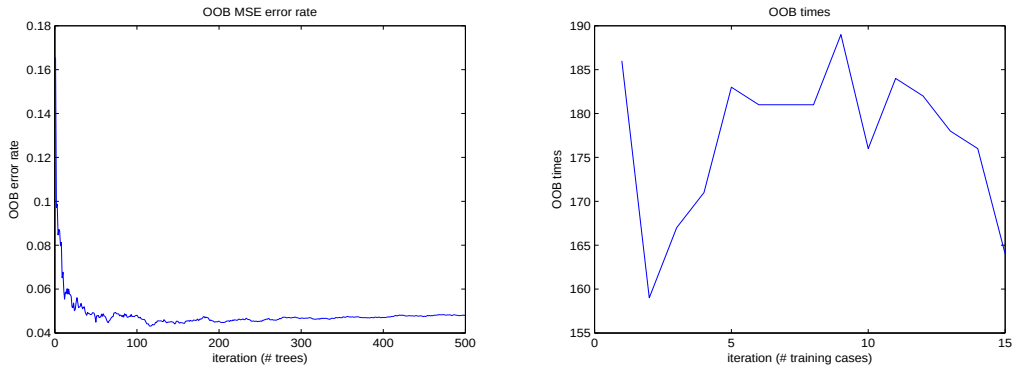
- During construction of each tree, an alternate bootstrap sample from the original data is used. Something like one-third of the cases from the bootstrap sample are left out and not used in the tree construction process.
- These OOB samples are put down the kth tree to obtain a value of regression. Using this process, a test set is acquired for each one case.
- At the end, suppose 'j' be the predicted value that is acquired by computing the average prediction value of forest, each time case 'n' was oob. The extent

of times 'j' is not equivalent to the actual value of 'n' averaged over all cases is called as the *out-of-bag error estimate*.

The RF prediction accuracy can be determined from these OOB data, by using the following formula:

$$OOB - MSE = \frac{1}{F}\sum_{i=1}^{F}(y_i - \bar{y}_{iOOB})^2 \tag{6.11}$$

where $\bar{y}_{iOOB}$ represents the average prediction value of ith observation from all trees for which this observation has been OOB. F denotes the no. of trees in the forest and y_i represents the actual value.



(a) (b)

Figure 6.6: OOB MSE Error Rate and Number of Times Out Of Bag Occurs

Figure 6.6a displays the OOB error rate obtained for different number of trees used in the forest. From the figure, it may be observed that during initial phase (while the number of trees used are less), the OOB error rate obtained is maximum. At the same time, with the increment of the amount of trees utilized within the forest, the OOB error rate converges to minimum value. After some period of time, OOB error rate remains constant. Figure 6.6b displays the number of times, cases are out of bag for all training attributes. In this case, fifteen number of training attributes are used.

Proximities

Proximity is one of the important data objects while calculating effort using RF technique. It measures the frequency of ending up the unique pairs of training samples in the same terminal node. It also helps in filling up the missing data in the dataset and calculating number of outliers.

Figure 6.7 describes the proximity value generated using random forest technique. The elements of matrix of size '15 $\times$ 15' have been used for generating the above

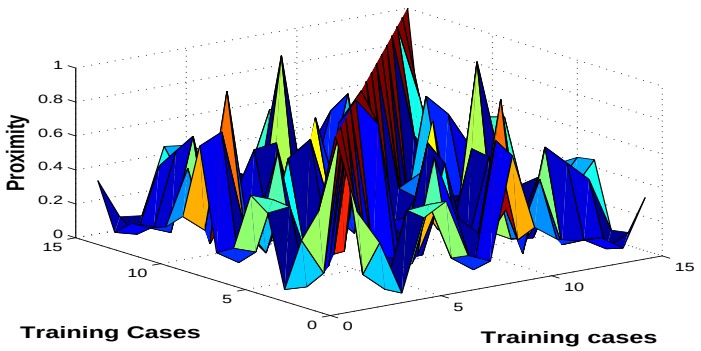


Figure 6.7: Proximity

figure. From the figure, it is observed that, for diagonal elements, the proximity value is maximum (equals to one). But for all other elements, the proximity value is less than one. The symmetric portion adjacent to diagonal area represents other elements proximity values.

Originally, a N × N matrix is formed by the proximities. Once a tree is developed, all the data i.e., training data and out-of-bag data are put down the tree. Its proximities should be increased by one, if it is found that two cases are in the same terminal node. Finally, the normalized values of the proximities are obtained by dividing with the number of trees.

Complexity

In the proposed approach, 500 number of trees are taken into consideration for implementing RF technique. In the usual tree growing algorithm, all descriptors are tested for their splitting performance at each node; while Random Forest only tests 'm' try of the descriptors. Since 'm' try is typically very small, the search is very fast.

To get the right model complexity for optimal prediction strength, some pruning is usually done via cross validation for a single decision tree. This process can take up a significant portion of the computations. RF, on the other hand, does not perform any pruning at all. It is observed that in cases where there are an excessively large number of descriptors, RF can be trained in less time than a single decision tree. Hence, the RF algorithm can be very efficient.

Outlier

The cases which are expelled from the principal group of data and whose proximities to all different cases in the data mostly small are defined as *Outliers*. The concept of outliers can be revised by defining outliers relative to corresponding cases. In this

way, an outlier is a case whose proximities to all different cases are little. The average proximity is specified as:

$$\bar{P}(n) = \sum_{1}^{N} prox^2(n,k) \tag{6.12}$$

where n and k denote a training case in the regression and N represents the total no. of training cases in the forest. The raw outlier measure for case n is specified as:

$$nsample/\bar{P}(n) \tag{6.13}$$

The result of raw outlier measure inversely depends on the average proximities. The average of these raw measures and their deviations from the average are ascertained for each case. The final outlier measure is obtained by subtracting the average from every raw measure, and afterwards dividing it by absolute deviation.

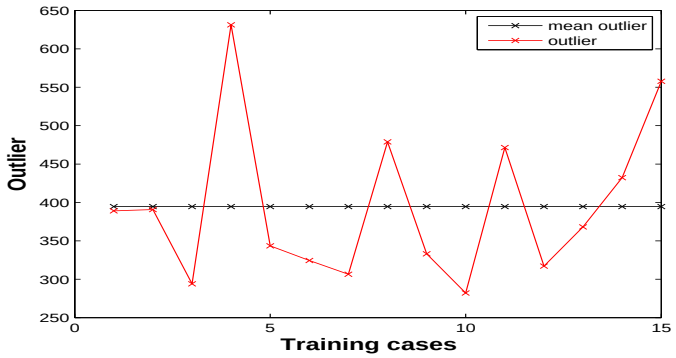


Figure 6.8: Outlier

Figure 6.8 describes the outlier value generated using random forest technique for 15 number of training cases. The outlier value is dependent on the proximity value generated using RF technique, which means that the outlier value is higher for lower proximity value and vice versa. Figure 6.8 displays the deviation of outlier value from the mean outlier. The training cases for which the outlier value is higher, generate the predicted effort value deviated more from actual effort value.

This deviation is clearly visible from Figure 6.9. It displays the final effort estimation model obtained using RF technique. The figure also shows the variation of actual effort from the predicted result obtained using RF technique.

6.4.2 Model Design using Various SVR Kernel Methods

After partitioning data into learning set and validation set, the model selection for ϵ and γ is performed using 5-fold cross validation process. In this method, to perform model selection, the ϵ and γ values are varied over a range. The γ value ranges

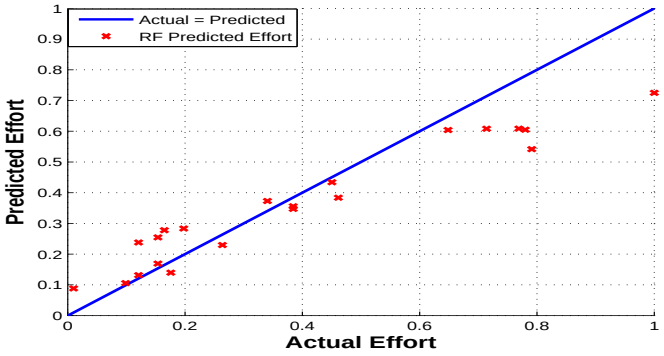


Figure 6.9: Actual vs. Predicted Graph obtained using Random Forest Technique for SPA

from 2^{-7} to 2^{7} and ϵ value ranges from 0 to 5. Hence, ninety number of models are generated to perform model selection operation.

Table 6.5: Validation Errors Obtained Using SVR Linear Kernel for SPA

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	**0.0190**	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-6}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-5}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-4}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-3}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-2}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-1}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046
2^{0}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046
2^{1}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046
2^{2}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046
2^{3}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046
2^{4}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046
2^{5}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046
2^{6}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046
2^{7}	0.0190	0.1046	0.1046	0.1046	0.1046	0.1046

Table 6.6: Validation Errors Obtained Using SVR Polynomial Kernel for SPA

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.0788	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-6}	0.0788	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-5}	0.0788	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-4}	0.0788	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-3}	0.0786	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-2}	0.0775	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-1}	0.0686	0.1046	0.1046	0.1046	0.1046	0.1046
2^{0}	**0.0378**	0.1046	0.1046	0.1046	0.1046	0.1046
2^{1}	0.0392	0.1046	0.1046	0.1046	0.1046	0.1046
2^{2}	0.0634	0.1046	0.1046	0.1046	0.1046	0.1046
2^{3}	0.0634	0.1046	0.1046	0.1046	0.1046	0.1046
2^{4}	0.0634	0.1046	0.1046	0.1046	0.1046	0.1046
2^{5}	0.0634	0.1046	0.1046	0.1046	0.1046	0.1046
2^{6}	0.0636	0.1046	0.1046	0.1046	0.1046	0.1046
2^{7}	0.0645	0.1046	0.1046	0.1046	0.1046	0.1046

Tables 6.5 and 6.6 show the validation error of ninety numbers of models generated for SPA using SVR linear kernel and SVR polynomial kernel respectively based on the value of ϵ and γ for 21 project dataset. For SVR Linear kernel, *0.0190* value has been chosen as the minimum validation error. Based on the minimum validation error, the best model is C = 0.78121, $\gamma = 0.0078125$ and $\epsilon = 0$. For SVR Polynomial kernel, *0.0378* value has been chosen as the minimum validation error. Based on the minimum validation error, the best model is C = 0.78121, $\gamma = 1$ and $\epsilon = 0$.

Tables 6.7 and 6.8 show the validation error of ninety numbers of models generated for SPA using SVR RBF kernel and SVR Sigmoid kernel respectively based on the value of ϵ and γ for 21 project dataset. For SVR RBF kernel, *0.0167* value has been chosen as the minimum validation error. Based on the minimum validation error, the best model is C = 0.78121, $\gamma = 1$ and $\epsilon = 0$. For SVR Sigmoid kernel, *0.0202* value has been chosen as the minimum validation error. Based on the minimum validation

Table 6.7: Validation Errors Obtained Using SVR RBF Kernel for SPA

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.0740	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-6}	0.0694	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-5}	0.0608	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-4}	0.0497	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-3}	0.0350	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-2}	0.0245	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-1}	0.0207	0.1046	0.1046	0.1046	0.1046	0.1046
2^{0}	**0.0167**	0.1046	0.1046	0.1046	0.1046	0.1046
2^{1}	0.0178	0.1046	0.1046	0.1046	0.1046	0.1046
2^{2}	0.0185	0.1046	0.1046	0.1046	0.1046	0.1046
2^{3}	0.0204	0.1046	0.1046	0.1046	0.1046	0.1046
2^{4}	0.0266	0.1046	0.1046	0.1046	0.1046	0.1046
2^{5}	0.0310	0.1046	0.1046	0.1046	0.1046	0.1046
2^{6}	0.0371	0.1046	0.1046	0.1046	0.1046	0.1046
2^{7}	0.0365	0.1046	0.1046	0.1046	0.1046	0.1046

Table 6.8: Validation Errors Obtained Using SVR Sigmoid Kernel for SPA

	$\epsilon = 0$	1	2	3	4	5
$\gamma = 2^{-7}$	0.0764	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-6}	0.0740	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-5}	0.0694	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-4}	0.0607	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-3}	0.0495	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-2}	0.0347	0.1046	0.1046	0.1046	0.1046	0.1046
2^{-1}	0.0245	0.1046	0.1046	0.1046	0.1046	0.1046
2^{0}	**0.0202**	0.1046	0.1046	0.1046	0.1046	0.1046
2^{1}	0.0214	0.1046	0.1046	0.1046	0.1046	0.1046
2^{2}	0.1274	0.1046	0.1046	0.1046	0.1046	0.1046
2^{3}	0.4728	0.1046	0.1046	0.1046	0.1046	0.1046
2^{4}	1.3989	0.1046	0.1046	0.1046	0.1046	0.1046
2^{5}	1.2401	0.1046	0.1046	0.1046	0.1046	0.1046
2^{6}	0.7182	0.1046	0.1046	0.1046	0.1046	0.1046
2^{7}	0.6623	0.1046	0.1046	0.1046	0.1046	0.1046

error, the best model is C = 0.78121, $\gamma = 1$ and $\epsilon = 0$. Based on model parameters value, the model has been again trained and tested using training and testing data set respectively to estimate the effort.

The proposed model generated using the SVR linear, polynomial, RBF and sigmoid kernel for SPA using 21 project dataset have been plotted as shown in Figures 6.10a, 6.10b, 6.10c and 6.10d respectively. These figures display the actual effort and the predicted effort obtained for SPA using the four SVR kernel methods taking into consideration 21 project dataset.

SVR Linear Kernel Result:

Param: -s 3 -t 0 -c 0.78121 -g 0.0078125 -p 0

* Mean Squared Error (MSE_TEST) = 0.0130

* Squared correlation coefficient = 0.9068

SVR Polynomial Kernel Result:

Param: -s 3 -t 1 -c 0.78121 -g 1 -p 0

* Mean Squared Error (MSE_TEST) = 0.0541

* Squared correlation coefficient = 0.4920

SVR RBF Kernel Result:

Param: -s 3 -t 2 -c 0.78121 -g 1 -p 0

* Mean Squared Error (MSE_TEST) = 0.0027

* Squared correlation coefficient = 0.9687

SVR Sigmoid Kernel Result:

Param: -s 3 -t 3 -c 0.78121 -g 1 -p 0

* Mean Squared Error (MSE_TEST) = 0.0122

* Squared correlation coefficient = 0.8903

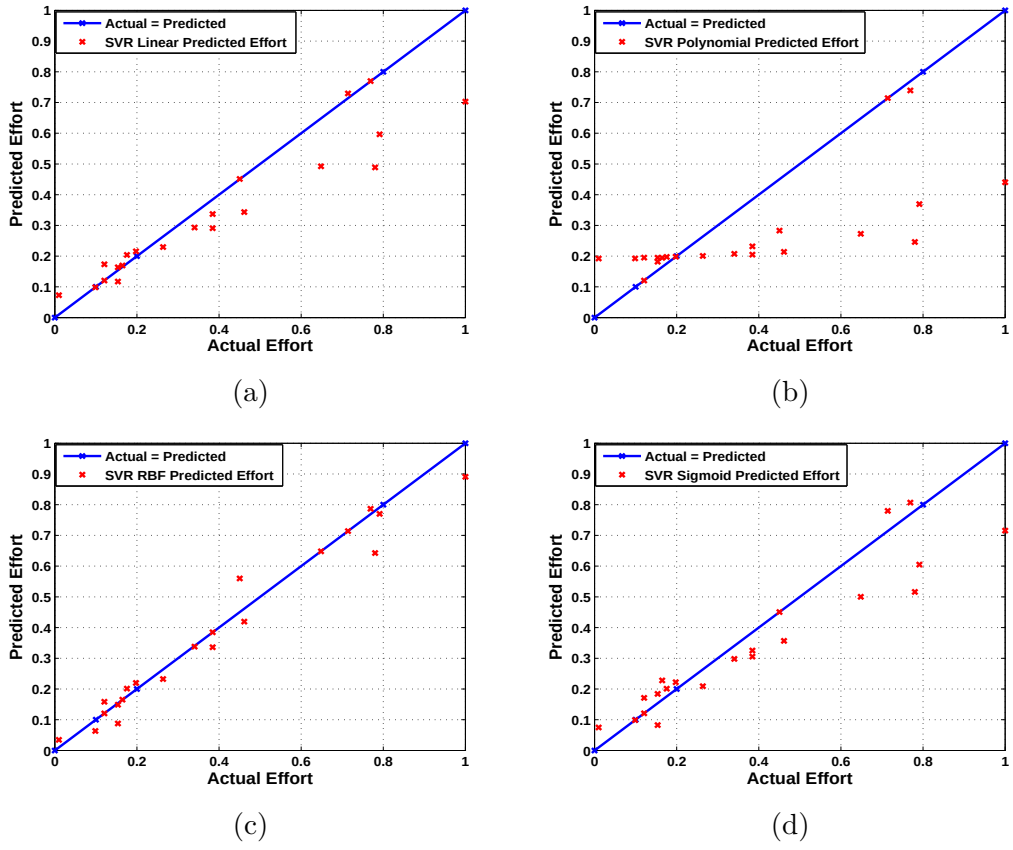


Figure 6.10: SVR Linear, Polynomial, RBF and Sigmoid Kernel based Agile Software Effort Estimation Model using SPA

In these graphs, it is observed that the data points are very less dispersed than the regression line. Hence the correlation is higher. While comparing the dispersion of data points from the predicted model in the above graphs, it is noted that the data points are less dispersed for SVR RBF kernel based model than other models. Hence, this model exhibits less error values and higher prediction accuracy value. The *squared correlation coefficient*(r^2) is also known as the *coefficient of determination*. It is one of the suitable means for evaluating the strength of a relationship. From the data associated with output, it may be noted that the *squared correlation coefficient* value for SVR Linear, RBF and Sigmoid kernel is very high (greater than 0.9). Hence it can be concluded that there is a strong positive correlation exists between the story point and the predicted effort required to develop the software i.e., a minor change in the class point value results in significant change in the predicted effort value.

6.5 Comparative Analysis

While using the MMER and PRED in evaluation, significantly convincing results are implied by lower value of the MMER and higher value of the PRED. Various algorithms such as stochastic gradient boosting (SGB) algorithm and Decision Tree Forests algorithm exhibit functional similarity, because SGB creates a tree ensemble, and also uses randomization during the creations of the trees. It creates a series of trees, and the prediction accuracy is calculated by feeding the result obtained from one tree to the next tree in the series. However, RF builds trees in parallel and also uses voting method on the prediction.

Table 6.9: Comparison of Proposed Results with Existing work

	Proposed Work	**Regression [97]**
PRED(25)	80.9524	57.14

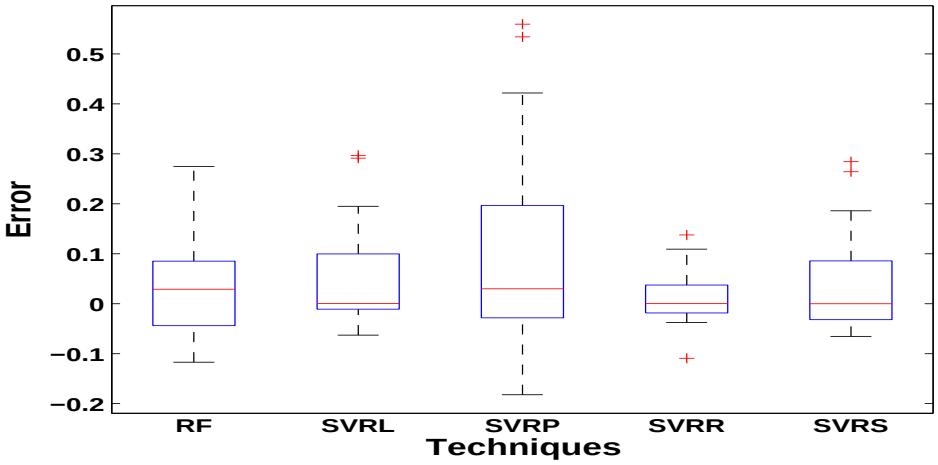
In Table 6.9, results obtained have been compared with the existing work in literature. It can be seen that both the random forest and SVR kernel models outperformed over the existing method by [97].

Table 6.10: Comparison of MMER and PRED Values between the RF and four SVR Kernel Techniques

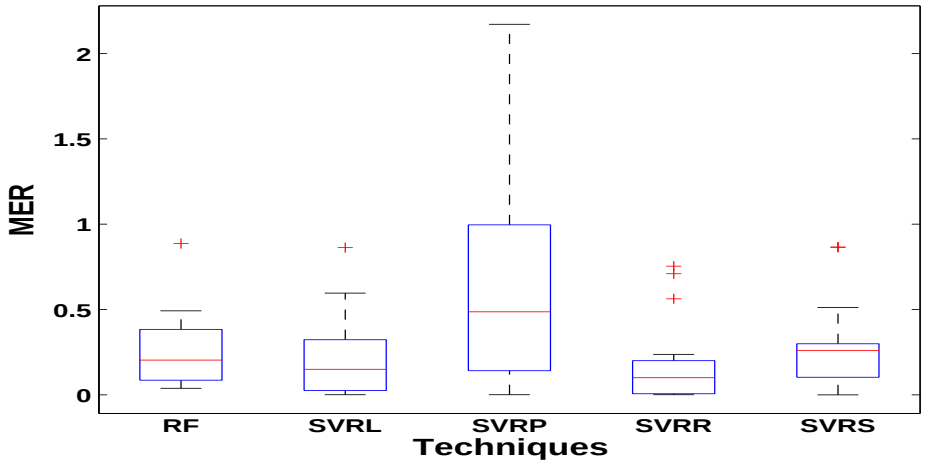
	MMER	**MdMER**	**PRED(25)**
Random Forest (RF)	**0.2516**	**0.2033**	**66.6667**
SVR Linear Kernel	**0.2186**	**0.1491**	**76.1905**
SVR Polynomial Kernel	**0.6040**	**0.4862**	**38.0952**
SVR RBF Kernel	**0.1667**	**0.1005**	**80.9524**
SVR Sigmoid Kernel	**0.2611**	**0.2595**	**71.4286**

At the point when utilizing the MMER, and PRED in assessment, good outcomes are entailed by lower estimations of MMER and higher estimations of PRED. Table 6.10 demonstrates the comparison of MMER and PRED values for the SVR and RF techniques. This comparative study helps in accurately assessing the performance obtained using these five techniques and proves that the results obtained using SVR RBF kernel technique-based effort estimation model outperforms the results obtained using other existing models.

Figures 6.11a and 6.11b display the box plot of Error and MER values for SPA using 21 project dataset respectively. These figures help to illustrate the spread and



(a)



(b)

Figure 6.11: Boxplot of Error and MER Values for SPA

differences of samples, with the help of their corresponding error values generated using RF, and different SVR kernel methods.

Table 6.11: Comparison of Effect Size Test of Proposed Models for SPA using 21 Project Dataset

	Cohen's d Effect Size	Glass's Δ
RF vs. SVR-Linear	0.1023	0.0965
RF vs. SVR-Polynomial	0.4539	0.4943
RF vs. SVR-RBF	0.0847	0.0736
RF vs. SVR-Sigmoid	0.0566	0.0523

In order to affirm the robustness of the proposed models, the effect size test [131]

such as Cohen's d and Glass's Δ test between diverse proposed models are processed considering absolute residuals as demonstrated in Table 6.11 for SPA using 21 project. From the results provided in Table 6.11, it is evident that the effect size is mostly small, except RF vs. SVR Polynomial Kernel, where it is approaching towards medium effect.

6.6 Summary

The story point approach is considered by several authors and used for an effective effort estimation of softwares developed using agile methodologies, especially scrum. In this chapter, an attempt has been made to apply story point approach for agile software effort estimation purpose. The results of story point approach are further improved by using random forest and four SVR kernel models. The obtained results are then validated and compared with the existing result obtained from Zia et al. [97]. The results demonstrate that the SVR RBF technique outperforms other used machine learning techniques for considered dataset. The computations for above procedure have been implemented, and outputs have been generated using MATLAB.

Chapter 7

Conclusion

It is observed in literature that analysts and practitioners have proposed several techniques for software effort estimation purpose. However, the CP, UCP and SPA are one of the effort estimation models which are used because of their simplicity, fastness and accuracy to a certain degree. The research contributions, conclusive remarks taking into account of the experimental research work carried out are incorporated into this chapter alongside the scope for future work.

7.1 Research Contributions

Chapter 1 presented the overview of the research work in the present thesis. It clarifies the current issue, and the proposed models by applying distinctive machine learning techniques. The chapter demonstrates the motivation behind the work, and the detailed explanation of the research objective proposed. At last, the chapter gave a brief layout of the thesis.

Chapter 2 discusses on the literature survey work done in the area of software effort estimation. The survey for the most part abridged the work done by various analysts and professionals in this research area. The entire chapter has been divided into six subsections. In the first section, emphasis was laid on the basic software effort estimation techniques such as function point, COCOMO, expert technique, analogy based software effort estimation etc. This survey was useful to presume that, among the diverse accessible techniques for software effort estimation, none of them proved to be the most effective one for estimating the effort of object-oriented softwares at an early stage during the software development life cycle. It was also observed that machine learning methods were more often used in most of the studies done by authors as noted in literature to improve the performance of prediction when compared with statistical approaches. It was further observed that a number of authors have also used datasets available in PROMISE and NASA repositories.

In the second and third section, survey is carried out on estimating the effort of software developed using object oriented methodology with the help of class point and use case point approach. From the analysis, it is observed that a number of authors have worked on class point and use case point approach. But the accuracy of their results on estimation can be improved by employing some other machine learning techniques using the same dataset.

The fourth section presents survey of articles dealing with effort estimation of web applications. The survey indicates that a number of authors worked on this area and applied various machine learning techniques over their dataset. But most of them are based on Tukutuku dataset, which is not publicly available [158–162]. Also, the accuracy of their estimation can also be improved by employing certain other machine learning techniques over their considered dataset.

The fifth section discusses on articles dealing with agile software development and its effort estimation procedure. From the survey, it was observed that effort estimation of softwares developed using agile methodology is a very tedious task and needs proper analysis due to its changeability feature. Hence, there is a necessity of developing proper estimation techniques for agile software development process.

The sixth section emphasizes on presenting few research works dealing with applying machine learning techniques for software effort estimation process and their corresponding implications.

Chapter 3 identifies various models designed for estimating effort of object oriented softwares using class point approach and applying different machine learning techniques over the CPA dataset. In the first phase of this chapter, the class point approach used for software effort estimation process is explained. The class point model is one of the various effort estimation models and preferred for projects developed using object-oriented technologies. This model helps in calculating the effort during an early stage of the software development life cycle. The second phase of this chapter focuses on applying the SGB and four SVR kernel techniques in order to optimize the parameters of class point approach. The generated results are compared with existing as well as among these proposed results. The results show that the SVR RBF Kernel-based effort estimation model possesses lower RMSE, MAE, MMRE, MMER and higher prediction accuracy. Hence it can be concluded that the effort estimation using the SVR RBF kernel-based model provides more accurate results than the results obtained using other machine learning techniques.

Chapter 4 emphasized on the models designed for estimating effort of object oriented softwares using use case point approach and applying different machine learning techniques over the UCP dataset. In the first phase of this chapter, the UCP approach used for software effort estimation process is explained. The UCP is

one of the effort estimation models, which is used because of its simplicity, fastness and accurateness to a certain degree. In the second phase of this chapter, the parameters of use case point approach have been optimized by applying the RF and various SVR kernel techniques. The RF and SVR kernel techniques are ensemble learning methods for regression, which combine the results from different models of similar type or different and gives result which is usually better than the result from other individual models. Hence, the use of UCP approach implemented using the RF and various SVR kernel techniques make advances in making better software effort estimation. The generated results are then compared with the results obtained from the LLR technique. After analyzing the results, it is found out that the effort estimation model developed using SVR RBF kernel technique provides less value of MMER and higher values of PRED. Consequently, it can be inferred that the effort estimation model developed on the considered dataset using SVR RBF kernel technique will give more exact results than other machine learning techniques based effort estimation models.

Chapter 5 discusses on the set of models designed for effort estimation of web applications and applying different machine learning techniques over the web dataset collected from ISBSG Release 12 repository. In the first phase of this chapter, the ISBSG dataset is collected and processed in order to extract the relevant fields which are going to be used for effort estimation process. The ISBSG Relese 12 repository contains dataset related to three categories of projects such as new development type, enhancement type and re-development type. In this phase out of all the projects, only new and enhanced software projects are considered and their corresponding data are taken as input to the different machine learning techniques employed. Both the new and enhanced type of web projects are further divided into three categories based on the value of their corresponding productivity factors. In the second phase of this chapter, different machine learning techniques such as the Decision Tree, Stochastic Gradient Boosting, Random Forest and four SVR kernel techniques have been employed in order to enhance their prediction accuracy of the web effort estimation model. It is observed that in case of both new and enhanced web projects, the parameters such as minimal error and higher prediction accuracy are achieved by applying SVR RBF kernel technique-based effort estimation model for all the six considered types of project datasets.

Chapter 6 presents various models designed for effort estimation of softwares developed using agile methodology i.e., Scrum by considering story point approach and applying different machine learning techniques over scrum based agile dataset. In the first phase of this chapter, different agile software development process has been discussed and the use of story point approach for scrum based agile software

development has been analyzed. The story point approach is a popular method used for agile software's effort estimation. In this approach, first the total number of story points and project velocity are used to estimate the effort involved in developing an agile software product. In the second phase of this chapter, the accuracy of story point approach have been further improved by using random forest and four SVR kernel models. At the end of the study, the generated results are compared among themselves as well as with the existing result obtained from Zia et al. [97]. The results demonstrate that the SVR RBF technique provides lower estimates of MMER and higher estimates of prediction accuracy. Consequently, it could be inferred that effort estimation utilizing the SVR RBF technique outperforms other machine learning techniques.

7.2 Concluding Remarks

The overall conclusions that can be inferred from the research work exhibited in this thesis is that the different findings obtained are surely be advantageous for the analysts, experts, and the product experts, in light of the fact that the CPA and UCP are used basically for object-oriented software and optimized by employing ML techniques to provide more accurate estimation result. For handling web-based applications, the ISBSG release 12 dataset are employed and then results are also optimized using different ML techniques to predict the effort more accurately. Similarly, SPA is one of the effort estimation models that can be applied to estimate the effort required to develop software using agile methodology. The obtained results are optimized using various ML techniques in order to improve the accuracy of the estimated effort value. Out of all the techniques used in different chapters, in most of the cases SVR polynomial performs poorly. Each SVR kernel is based on certain kernel function. Any operation for that kernel is done with the help of their corresponding kernel function. RBF kernel uses exponential function, where sigmoid kernel uses sigmoid function. Linear kernel is more preferable for linearly separable data. Hence, by analyzing the obtained results, it is observed that different results (error and prediction accuracy values) are obtained using different kernels and the result obtained utilizing SVR RBF kernel based effort estimation model outperform the results obtained from other models for CPA, UCP, SPA as well as for web applications. The computations for above methodologies were executed, and results were obtained using MATLAB.

There are certain limitations of the implementation. The quality of the research gets enhanced, if the output is based on the empirical analysis, where sufficient past data are fed as input to the study. The theoretician faced the problem because

sufficient historical data for the effort estimation purpose are not available to them. In industries, the developers are under so much pressure that they give less emphasis on the correctness of the estimation process adopted by them. With this limitation of non-availability of right kind of historical data, an attempt has been made to carry out research based on software effort estimation. The data collected from various research articles are some times too small and are not exhaustive. ISBSG dataset contains several missing entries, which lead to discarding of the record. Due to this, there was a considerable reduction in the size of dataset. If the missing entries are filled using some missing value handling techniques, the quality of results may improve. Moreover, due to small size of dataset i.e., of size in a matrix form of 18×4 in case of redevelopment type of web-based projects, no. of data left for testing are very less. Thus optimal accuracy of the model's performance cannot be guaranteed. Hence, this type of projects are not considered for implementation. If more number of dataset related to redevelopment type of projects would have been available, it could have added more flavor to quality of result.

All the models proposed for agile software effort estimation have been developed by assuming that the initial project velocity value is given. This value is taken from the past projects developed by the same team in similar working conditions. But when a team is new, the company may not be having any past record for it. In that case, no clear assignment to initial project velocity can be done. The dataset collected from [97] for agile software effort estimation purpose does not provide any information about the type of projects taken into consideration for this study. For the obtained results to be valid for the general software engineering paradigm, the work is desired to be based on data, which covers all categories of software developed using different agile methodologies.

7.3 Future Scope of Work

Nobody can unwrap the future. The future is a state based on the series of events that have taken place since the initial state. Over the long haul, the effort put in are of concern. Triggered by this view point, all the work reported in this thesis work will prompt augmentation that will improve their impact in the particular area of work.

Extension to this procedure can be made by applying ensembles of machine learning techniques for the software development effort estimation purpose in order to assess the performance achieved by the techniques considered in earlier mentioned chapters. Also, in order to estimate effort required to complete the non-functional requirements, a procedure called Software Non-functional Assessment Procedure

(SNAP) point [163] approach can be employed. SNAP point approach is complement to the function point sizing. Function points measure the functional requirements by sizing the data flow through a software application; whereas SNAP measures the non-functional requirements. Various ML techniques can also be applied over the SNAP point dataset in order to improve the accuracy of effort estimation.

New methodologies could be derived for estimation the effort of soft- ware based on Service Oriented Architecture (SOA) [164, 165] and Cloud. Also, as Aspect-Oriented Programming (AOP) [166, 167] and Feature-Oriented Programming (FOP) [168, 169] concepts are very popular nowadays, hence new approaches could be identified in order to estimate the effort required to develop a software based on AOP and FOP concepts.

Bibliography

[1] Robert N Charette. Why software fails. *IEEE spectrum*, 42(9):36, 2005.

[2] Sanchoy K Das, Pradeep Yedlarajiah, and Raj Narendra. An approach for estimating the end-of-life product disassembly effort and cost. *International Journal of Production Research*, 38(3):657–673, 2000.

[3] STANDISH GROUP et al. Chaos manifesto. *The Standish Group*, 2013.

[4] Onur Demirörs and Çiğdem Gencel. A comparison of size estimation techniques applied early in the life cycle. In *Software Process Improvement*, pages 184–194. Springer, 2004.

[5] Ian Sommerville and Gerald Kotonya. *Requirements engineering: processes and techniques*. John Wiley & Sons, Inc., 1998.

[6] D Eck, B Brundick, T Fettig, J Dechoretz, and J Ugljesa. Parametric estimating handbook. *The International Society of Parametric Analysis (ISPA),*, 2009.

[7] Ali Bou Nassif. *Software Size and Effort Estimation from Use Case Diagrams Using Regression and Soft Computing Models*. PhD thesis, Western University, 2012.

[8] STANDISH GROUP et al. The chaos manifesto 2011. *The Standish Group International. EUA*, 2011.

[9] Donald J Reifer. Web development: estimating quick-to-market software. *IEEE software*, 17(6):57–64, 2000.

[10] ISBSG. The international software benchmarking standards group. `http://www.isbsg.org`, 2011.

[11] Ofer Morgenshtern, Tzvi Raz, and Dov Dvir. Factors affecting duration and effort estimation errors in software development projects. *Information and Software Technology*, 49(8):827–837, 2007.

[12] Adriano LI Oliveira. Estimation of software project effort with support vector regression. *Neurocomputing*, 69(13):1749–1753, 2006.

[13] Anna Corazza, Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, Federica Sarro, and Emilia Mendes. How effective is tabu search to configure support vector regression for effort estimation? In *Proceedings of the 6th international conference on predictive models in software engineering*, page 4. ACM, ACM, 2010.

[14] Anna Corazza, Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, and Emilia Mendes. Investigating the use of support vector regression for web effort estimation. *Empirical Software Engineering*, 16(2):211–243, 2011.

[15] Petrônio L Braga, Adriano LI Oliveira, and Silvio RL Meira. A ga-based feature selection and parameters optimization for support vector regression applied to software effort estimation. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1788–1792. ACM, 2008.

[16] James N Morgan and John A Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American statistical association*, 58(302):415–434, 1963.

[17] James N Morgan and Robert C Messenger. *THAID: A sequential analysis program for the analysis of nominal scale dependent variables.* Ann Arbor, Survey Research Center, Institute for Social Research, University of Michigan, 1973.

[18] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[19] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[20] Harris Drucker, Chris JC Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in neural information processing systems*, pages 155–161, 1997.

[21] Alex J Smola and Bernhard ölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[22] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[23] Tim Menzies, Zhihao Chen, Jairus Hihn, and Karen Lum. Selecting best practices for effort estimation. *Software Engineering, IEEE Transactions on*, 32(11):883–895, 2006.

[24] Ali Bou Nassif, Danny Ho, and Luiz Fernando Capretz. Towards an early software estimation using log-linear regression and a multilayer perceptron model. *Journal of Systems and Software*, 86(1):144–160, 2013.

[25] Mohammad Azzeh, Ali Bou Nassif, and Leandro L Minku. An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation. *Journal of Systems and Software*, 103:36–52, 2015.

[26] Jerry R Thomas, Walter Salazar, and Daniel M Landers. What is missing in p¡. 05? effect size. *Research quarterly for exercise and sport*, 62(3):344–348, 1991.

[27] B.A. Kitchenham, L.M. Pickard, S.G. MacDonell, and M.J. Shepperd. What accuracy statistics really measure [software estimation]. *Software, IEE Proceedings* -, 148(3):81–85, Jun 2001.

[28] Tron Foss, Erik Stensrud, Barbara Kitchenham, and Ingunn Myrtveit. A simulation study of the model evaluation criterion mmre. *Software Engineering, IEEE Transactions on*, 29(11):985–995, 2003.

[29] David J Sheskin. *Handbook of parametric and nonparametric statistical procedures.* crc Press, 2003.

[30] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[31] Graeme D Ruxton. The unequal variance t-test is an underused alternative to student's t-test and the mann–whitney u test. *Behavioral Ecology*, 17(4):688–690, 2006.

[32] Michael P Fay and Michael A Proschan. Wilcoxon-mann-whitney or t-test? on assumptions for hypothesis tests and multiple interpretations of decision rules. *Statistics surveys*, 4:1, 2010.

[33] Lee A Becker. Effect size (es). *Accessed on October*, 12(2006):155–159, 2000.

[34] Vigdis By Kampenes, Tore Dybå, Jo E Hannay, and Dag IK Sjøberg. A systematic review of effect size in software engineering experiments. *Information and Software Technology*, 49(11):1073–1086, 2007.

[35] Jacob Cohen. A power primer. *Psychological bulletin*, 112(1):155, 1992.

[36] Lawrence H. Putnam. A general empirical solution to the macro software sizing and estimating problem. *IEEE transactions on Software Engineering*, 4(4):345, 1978.

[37] Krishnakumar Pillai and VS Sukumaran Nair. A model for software development effort and cost estimation. *Software Engineering, IEEE Transactions on*, 23(8):485–497, 1997.

[38] Allan J Albrecht. Measuring application development productivity. In *Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium*, volume 10, pages 83–92, 1979.

[39] IFPUG IFPUG. Function point counting practices manual, release 4.2. *International Function Point Users Group, USA–IFPUG, Mequon, Wisconsin, USA*, 2004.

[40] Barry W Boehm et al. *Software engineering economics*, volume 197. Prentice-hall Englewood Cliffs (NJ), 1981.

[41] Robert T Hughes. Expert judgement as an estimating method. *Information and Software Technology*, 38(2):67–75, 1996.

[42] Norman Dalkey and Olaf Helmer. An experimental application of the delphi method to the use of experts. *Management science*, 9(3):458–467, 1963.

[43] Magne Jørgensen. Forecasting of software development work effort: Evidence on expert judgement and formal models. *International Journal of Forecasting*, 23(3):449–462, 2007.

[44] Martin Shepperd, Chris Schofield, and Barbara Kitchenham. Effort estimation using analogy. In *Proceedings of the 18th international conference on Software engineering*, pages 170–178. IEEE Computer Society, 1996.

[45] Fiona Walkerden and Ross Jeffery. An empirical study of analogy-based software effort estimation. *Empirical software engineering*, 4(2):135–158, 1999.

[46] Ali Idri, Fatima azzahra Amazal, and Alain Abran. Analogy-based software development effort estimation: A systematic mapping and review. *Information and Software Technology*, 58:206–230, 2015.

[47] Ali Idri, Fatima azzahra Amazal, and Alain Abran. Accuracy comparison of analogy-based software development effort estimation techniques. *International Journal of Intelligent Systems*, 31(2):128–152, 2016.

[48] Kjetil Molokken and Magen Jorgensen. A review of software surveys on software effort estimation. In *Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on*, pages 223–230. IEEE, IEEE, 2003.

[49] Magne Jørgensen. Practical guidelines for expert-judgment-based software effort estimation. *Software, IEEE*, 22(3):57–63, 2005.

[50] Ekrem Kocaguneli, Tim Menzies, and Emilia Mendes. Transfer learning in effort estimation. *Empirical Software Engineering*, 20(3):813–843, 2015.

[51] Peter A Whigham, Caitlin A Owen, and Stephen G Macdonell. A baseline model for software effort estimation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(3):20, 2015.

[52] Fernando González-Ladrón-de Guevara, Marta Fernández-Diego, and Chris Lokan. The usage of isbsg data fields in software effort estimation: A systematic mapping study. *Journal of Systems and Software*, 113:188–215, 2016.

[53] G. Costagliola, F. Ferrucci, G. Tortora, and G. Vitiello. Class point: an approach for the size estimation of object-oriented systems. *Software Engineering, IEEE Transactions on*, 31(1):52–74, 2005.

[54] W. Zhou and Q. Liu. Extended class point approach of size estimation for oo product. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 4, pages 117–122. IEEE, 2010.

[55] S. Kanmani, J. Kathiravan, S. Senthil Kumar, and M. Shanmugam. Neural network based effort estimation using class points for oo systems. In *Proceedings of the International Conference on Computing: Theory and Applications*, ICCTA '07, pages 261–266, Washington, DC, USA, 2007. IEEE Computer Society.

[56] S. Kim, W. Lively, and D. Simmons. An effort estimation by uml points in early stage of software development. *Proceedings of the International Conference on Software Engineering Research and Practice*, pages 415–421, 2006.

[57] S. Kanmani, Jayabalan Kathiravan, S. Senhil Kumar, and Mourougane Shanmugam. Class point based effort estimation of oo systems using fuzzy subtractive clustering and artificial neural networks. In *Proceedings of the 1st India software engineering conference*, ISEC '08, pages 141–142, New York, NY, USA, 2008. ACM.

[58] Aditi Kapoor and Parul Pandey. Fuzzy class point approach. *Int. J. on Recent Trends in Engineering & Technology*, 5(01):15–18, 2011.

[59] Ayman Issa, Mohammed Odeh, and David Coward. Software cost estimation using use-case models: A critical evaluation. In *Information and Communication Technologies, 2006. ICTTA'06. 2nd*, volume 2, pages 2766–2771. IEEE, IEEE, 2006.

[60] Ali Bou Nassif, Danny Ho, and Luiz Fernando Capretz. Regression model for software effort estimation based on the use case point method. In *2011 International Conference on Computer and Software Modeling*, volume 14, pages 106–110, 2011.

[61] Tomas Urbanek, Zdenka Prokopová, and Radek Silhavy. On the value of parameters of use case points method. In *Artificial Intelligence Perspectives and Applications*, pages 309–319. Springer, 2015.

[62] Ali Bou Nassif, Luiz Fernando Capretz, and Danny Ho. A regression model with mamdani fuzzy inference system for early software effort estimation based on use case diagrams. In *Third International Conference on Intelligent Computing and Intelligent Systems*, pages 615–620. IEEE, 2011.

[63] Ali Bou Nassif, Luiz Fernando Capretz, and Danny Ho. Estimating software effort based on use case point model using sugeno fuzzy inference system. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pages 393–398. IEEE, 2011.

[64] Ali Bou Nassif, Luiz Fernando Capretz, and Danny Ho. Estimating software effort using an ann model based on use case points. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 42–47. IEEE, IEEE, 2012.

[65] Ali Bou Nassif. Enhancing use case points estimation method using soft computing techniques. *Journal of Global Research in Computer Science*, 1(4), 2010.

[66] Ali Bou Nassif, Luiz Fernando Capretz, Danny Ho, and Mohammad Azzeh. A treeboost model for software effort estimation based on use case points. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 314–319. IEEE, IEEE, 2012.

[67] Meenakshi Saroha and Shashank Sahu. Tools & methods for software effort estimation using use case points modela review. In *Computing, Communication & Automation (ICCCA), 2015 International Conference on*, pages 874–879. IEEE, 2015.

[68] Radek Silhavy, Petr Silhavy, and Zdenka Prokopová. Algorithmic optimisation method for improving use case points estimation. *PloS one*, 10(11):e0141887, 2015.

[69] Kasi Periyasamy and Aditi Ghode. Cost estimation using extended use case point (e-ucp) model. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pages 1–5. IEEE, 2009.

[70] Bente Anda, Hege Dreiem, Dag IK Sjøberg, and Magne Jørgensen. Estimating software development effort based on use casesexperiences from industry. In *≪ UML ≫ 2001The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, pages 487–502. Springer, 2001.

[71] Sergey Diev. Use cases modeling and software estimation: applying use case points. *ACM SIGSOFT Software Engineering Notes*, 31(6):1–4, 2006.

[72] S Ajitha, TV Suresh Kumar, Evangelin D Geetha, and K Rajani Kanth. Neural network model for software size estimation using use case point approach. In *Industrial and Information Systems (ICIIS), 2010 International Conference on*, pages 372–376. IEEE, 2010.

[73] Mohammad Saber Iraji and Homayun Motameni. Object oriented software effort estimate with adaptive neuro fuzzy use case size point (anfusp). *International Journal of Intelligent Systems and Applications*, 4(6):14, 2012.

[74] Ali Bou Nassif, Luiz Fernando Capretz, and Danny Ho. Calibrating use case points. In *Companion Proceedings of the 36th International Conference on Software Engineering*, pages 612–613. ACM, 2014.

[75] Meenakshi Saroha and Shashank Sahu. Software effort estimation using enhanced use case point model. In *Computing, Communication & Automation (ICCCA), 2015 International Conference on*, pages 779–784. IEEE, 2015.

[76] Emilia Mendes, Nile Mosley, and Steve Counsell. Investigating web size metrics for early web cost estimation. *Journal of Systems and Software*, 77(2):157–172, 2005.

[77] Emilia Mendes. A comparison of techniques for web effort estimation. In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, pages 334–343. IEEE, 2007.

[78] Emilia Mendes and Nile Mosley. Bayesian network models for web effort prediction: a comparative study. *Software Engineering, IEEE Transactions on*, 34(6):723–737, 2008.

[79] Filomena Ferrucci, Carmine Gravino, Rocco Oliveto, Federica Sarro, and Emilia Mendes. Investigating tabu search for web effort estimation. In *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on*, pages 350–357. IEEE, IEEE, 2010.

[80] Sanaa Elyassami and Ali Idri. Applying fuzzy id3 decision tree for software effort estimation. *arXiv preprint arXiv:1111.0158*, 2011.

[81] Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, and Federica Sarro. Using web objects for development effort estimation of web applications: a replicated study. In *Product-Focused Software Process Improvement*, pages 186–201. Springer, 2011.

[82] Filomena Ferrucci, Emilia Mendes, and Federica Sarro. Web effort estimation: the value of cross-company data set compared to single-company data set. In *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*, pages 29–38. ACM, ACM, 2012.

[83] Erika Corona, Giulio Concas, Michele Marchesi, Gianluca Barabino, and Daniele Grechi. Effort estimation of web applications through web cmf objects. In *Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2012 Joint Conference of the 22nd International Workshop on*, pages 15–22. IEEE, IEEE, 2012.

[84] Ekrem Kocaguneli, Tim Menzies, and Jacky W Keung. On the value of ensemble effort estimation. *Software Engineering, IEEE Transactions on*, 38(6):1403–1416, 2012.

[85] Damir Azhar, Patricia Riddle, Eduardo Mendes, Nikolaos Mittas, and Lefteris Angelis. Using ensembles for web effort estimation. In *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*, pages 173–182. IEEE, IEEE, 2013.

[86] Olavo Matos, Luiz Fortaleza, Tayana Conte, and Emilia Mendes. Realising web effort estimation: a qualitative investigation. In *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, pages 12–23. ACM, ACM, 2013.

[87] Olavo Matos, Tayana Conte, and Emilia Mendes. Is there a place for qualitative studies when identifying effort predictors?: a case in web effort estimation. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, page 40. ACM, ACM, 2014.

[88] Ali Bou Nassif, Mohammad Azzeh, Luiz Fernando Capretz, and Danny Ho. Neural network models for software development effort estimation: a comparative study. *Neural Computing and Applications*, pages 1–13, 2015.

[89] Denis eke and Boris Milainovi. Early effort estimation in web application development. *Journal of Systems and Software*, 103(0):219 – 237, 2015.

[90] Giulio Barabino, Giulio Concas, Erika Corona, Daniele Grechi, Michele Marchesi, and Danilo Tigano. Web framework points: an effort estimation methodology for web application development using a content management framework. *Journal of Software: Evolution and Process*, pages n/a–n/a, 2015.

[91] Mridul Bhardwaj and Ajay Rana. Impact of size and productivity on testing and rework efforts for web-based development projects. *SIGSOFT Softw. Eng. Notes*, 40(2):1–4, April 2015.

[92] Leandro Minku, Federica Sarro, Emilia Mendes, and Filomena Ferrucci. How to make best use of cross-company data for web effort estimation? In *Empirical Software Engineering and Measurement (ESEM), 2015 ACM/IEEE International Symposium on*, pages 1–10. IEEE, 2015.

[93] Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, and Federica Sarro. Web effort estimation: Function point analysis vs. cosmic. *Information and Software Technology*, 72:90–109, 2016.

[94] Siobhan Keaveney and Kieran Conboy. Cost estimation in agile development projects. In *ECIS*, pages 183–197, 2006.

[95] Evita Coelho and Anirban Basu. Effort estimation in agile software development using story points. *development*, 3(7), 2012.

[96] Andreas Schmietendorf, Martin Kunz, and Reiner Dumke. Effort estimation for agile software development projects. In *5th Software Measurement European Forum*, pages 113–123, 2008.

[97] Ziauddin Khan Zia, Shahid Kamal Tipu, and Shahrukh Khan Zia. An effort estimation model for agile software development. *Advances in Computer Science and its Applications*, 2(1):314–324, 2012.

[98] Muhammad Usman, Emilia Mendes, Francila Weidt, and Ricardo Britto. Effort estimation in agile software development: A systematic literature review. In *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, pages 82–91. ACM, 2014.

[99] Peter Hearty, Norman Fenton, David Marquez, and Martin Neil. Predicting project velocity in xp using a learning dynamic bayesian network model. *Software Engineering, IEEE Transactions on*, 35(1):124–137, 2009.

[100] Rashmi Popli and Naresh Chauhan. Cost and effort estimation in agile software development. In *Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on*, pages 57–61. IEEE, 2014.

[101] Ishrar Hussain, Leila Kosseim, and Olga Ormandjieva. Approximation of cosmic functional size to support early effort estimation in agile. *Data & Knowledge Engineering*, 85:2–14, 2013.

[102] Alaa El Deen Hamouda. Using agile story points as an estimation technique in cmmi organizations. In *Agile Conference (AGILE), 2014*, pages 16–23. IEEE, 2014.

[103] Erdir Ungan, Numan Cizmeli, and Onur Demirors. Comparison of functional size based estimation and story points, based on effort estimation effectiveness in scrum projects. In *Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on*, pages 77–80. IEEE, 2014.

[104] Viljan Mahnic. A case study on agile estimating and planning using scrum. *Elektronika ir Elektrotechnika*, 111(5):123–128, 2011.

[105] V Mahnic and N Zabkar. Measuring progress of scrum-based software projects. *Elektronika ir Elektrotechnika*, 18(8):73–76, 2012.

[106] Sakshi Garg and Daya Gupta. Pca based cost estimation model for agile software development projects. In *Industrial Engineering and Operations Management (IEOM), 2015 International Conference on*, pages 1–7. IEEE, 2015.

[107] Valentina Lenarduzzi, Ilaria Lunesu, Martina Matta, and Davide Taibi. Functional size measures and effort estimation in agile development: A replicated study. In *Agile Processes, in Software Engineering, and Extreme Programming*, pages 105–116. Springer, 2015.

[108] Atef Tayh Raslan, Nagy Ramadan Darwish, and Hesham Ahmed Hefny. Towards a fuzzy based framework for effort estimation in agile software development. *International Journal of Computer Science and Information Security*, 13(1):37, 2015.

[109] Ricardo Britto, Emilia Mendes, and Jurgen Borstler. An empirical investigation on effort estimation in agile global software development. In *Global Software Engineering (ICGSE), 2015 IEEE 10th International Conference on*, pages 38–45. IEEE, IEEE, 2015.

[110] Alaa Sheta. Software effort estimation and stock market prediction using takagi-sugeno fuzzy models. In *Fuzzy Systems, 2006 IEEE International Conference on*, pages 171–178. IEEE, 2006.

[111] Sanaa Elyassami and A Idri. Evaluating software cost estimation models using fuzzy decision trees. *Recent Advances in Knowledge Engineering and Systems Science, WSEAS Press*, pages 243–248, 2013.

[112] JS Pahariya, Vadlamani Ravi, and Mahil Carr. Software cost estimation using computational intelligence techniques. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 849–854. IEEE, 2009.

[113] Ekrem Kocaguneli, Tim Menzies, and Jacky W Keung. Kernel methods for software effort estimation. *Empirical Software Engineering*, 18(1):1–24, 2013.

[114] Bilge Baskeles, Burak Turhan, and Ayşe Bener. Software effort estimation using machine learning methods. In *Computer and information sciences, 2007. iscis 2007. 22nd international symposium on*, pages 1–6. IEEE, 2007.

[115] Jianfeng Wen, Shixian Li, Zhiyong Lin, Yong Hu, and Changqin Huang. Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology*, 54(1):41–59, 2012.

[116] Mohammad Azzeh, Daniel Neagu, and Peter I Cowling. Analogy-based software effort estimation using fuzzy numbers. *Journal of Systems and Software*, 84(2):270–284, 2011.

[117] Emilia Mendes, Veronica Taquete Vaz, and Fernando Muradas. An expert-based requirements effort estimation model using bayesian networks. In *Software Quality. The Future of Systems-and Software Development*, pages 79–93. Springer, 2016.

[118] M. Carbone and G. Santucci. Fast&&serious: a uml based metric for effort estimation. In *Proceedings of the 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'02)*, pages 313–322, 2002.

[119] FJ Heemstra and RJ Kusters. Function point analysis: Evaluation of a software cost estimation model. *European Journal of Information Systems*, 1(4):229–237, 1991.

[120] Hareton Leung and Zhang Fan. Software cost estimation. *Handbook of Software Engineering, Hong Kong Polytechnic University*, 2002.

[121] J.E. Matson, B.E. Barrett, and J.M. Mellichamp. Software development cost estimation using function points. *Software Engineering, IEEE Transactions on*, 20(4):275–287, 1994.

[122] Vladimir Cherkassky and Filip M Mulier. *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007.

[123] K. Srinivasan and D. Fisher. Machine learning approaches to estimating software development effort. *Software Engineering, IEEE Transactions on*, 21(2):126–137, Feb 1995.

[124] Nasib S Gill and Sunil Sikka. New complexity model for classes in object oriented system. *ACM SIGSOFT Software Engineering Notes*, 35(5):1–7, 2010.

[125] G Costagliola, F Ferrucci, G Tortora, and G Vitiello. Towards a software size metrics for object-oriented systems. *Proc. AQUIS*, 98:121–126, 1998.

[126] Shashank Mouli Satapathy, Mukesh Kumar, and Santanu Kumar Rath. Class point approach for software effort estimation using soft computing techniques. In *Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on*, pages 178–183. IEEE, 2013.

[127] Shashank Mouli Satapathy and Santanu Kumar Rath. Class point approach for software effort estimation using various support vector regression kernel methods. In *Proceedings of the 7th India Software Engineering Conference*, page 4. ACM, 2014.

[128] Shashank Mouli Satapathy, Barada Prasanna Acharya, and Santanu Kumar Rath. Class point approach for software effort estimation using stochastic gradient boosting technique. *ACM SIGSOFT Software Engineering Notes*, 39(3):1–6, 2014.

[129] Shashank Mouli Satapathy, Barada Prasanna Acharya, and Santanu Kumar Rath. Early stage software effort estimation using random forest technique based on optimized class point approach. *INFOCOMP Journal of Computer Science*, 13(2):22–33, 2014.

[130] Shashank Mouli Satapathy, Mukesh Kumar, and Santanu Kumar Rath. Fuzzy-class point approach for software effort estimation using various adaptive regression methods. *CSI transactions on ICT*, 1(4):367–380, 2013.

[131] Martin Shepperd and Steve MacDonell. Evaluating prediction systems in software project estimation. *Information and Software Technology*, 54(8):820–827, 2012.

[132] SG MacDonell and AR Gray. A comparison of modeling techniques for software development effort prediction. In *Proceedings of fourth International Conference on Neural Information Processing*, pages 869–872, Dunedin, New Zealand, 1998. Springer-Verlag.

[133] Kevin Strike, Khaled El Emam, and Nazim Madhavji. Software cost estimation with incomplete data. *Software Engineering, IEEE Transactions on*, 27(10):890–908, 2001.

[134] Mel Damodaran and A Washington. Estimation using use case points. *Computer Science Program. Texas–Victoria: University of Houston. Sd*, 2002.

[135] Gustav Karner. Resource estimation for objectory projects. *Objective Systems SF AB*, 17, 1993.

[136] Shashank Mouli Satapathy, Barada Prasanna Acharya, and Santanu Kumar Rath. Early stage software effort estimation using random forest technique based on use case points. *IET Software*, 10(1):10–17, 2016.

[137] Shashank Mouli Satapathy and Santanu Kumar Rath. Use case point approach based software effort estimation using various support vector regression kernel methods. *arXiv preprint arXiv:1401.3069*, 2014.

[138] Kirsten Ribu. Estimating object-oriented software projects with use cases. Master of science thesis, Department of Informatics, University of Oslo, November 2001.

[139] RandomForest-MATLAB. Random forest (regression, classification and clustering) implementation for matlab (and standalone). `https://code.google.com/p/randomforest-matlab/`, February 2010.

[140] Gennaro Costagliola, Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, Genoveffa Tortora, and Giuliana Vitiello. Effort estimation modeling techniques: a case study for web applications. In *Proceedings of the 6th international conference on Web engineering*, pages 9–16. ACM, ACM, 2006.

[141] Edilson JD Candido and Rosely Sanches. Estimating the size of web applications by using a simplified function point method. In *WebMedia and LA-Web, 2004. Proceedings*, pages 98–105. IEEE, IEEE, 2004.

[142] Damir Azhar, Emilia Mendes, and Patricia Riddle. A systematic review of web resource estimation. In *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*, pages 49–58. ACM, ACM, 2012.

[143] Adam Trendowicz, Jürgen Münch, and Ross Jeffery. State of the practice in software effort estimation: a survey and literature review. In *Software Engineering Techniques*, pages 232–245. Springer, 2011.

[144] Qiang Ye, Ziqiong Zhang, and Rob Law. Sentiment classification of online reviews to travel destinations by supervised machine learning approaches. *Expert Systems with Applications*, 36(3):6527–6535, 2009.

[145] Sotiris B Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31(3):249–268, 2007.

[146] Aik Choon Tan and David Gilbert. An empirical comparison of supervised machine learning techniques in bioinformatics. In *Proceedings of the First Asia-Pacific bioinformatics conference on Bioinformatics 2003-Volume 19*, pages 219–222. Australian Computer Society, Inc., ACM, 2003.

[147] International Software Benchmarking Standard Group. ISBSG, ISBSG dataset release 12. `http://www.isbsg.org/`, 2013.

[148] Shashank Mouli Satapathy and Santanu Kumar Rath. Effort estimation of web-based applications using machine learning techniques. In *Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on*, pages 978–984. IEEE, 2016.

[149] Martin Fowler and Jim Highsmith. The agile manifesto. *Software Development*, 9(8):28–35, 2001.

[150] David Cohen, Mikael Lindvall, and Patricia Costa. An introduction to agile methods. *Advances in Computers*, 62:1–66, 2004.

[151] Muhammad Usman, Emilia Mendes, and Jürgen Börstler. Effort estimation in agile software development: a survey on the state of the practice. In *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, page 12. ACM, ACM, 2015.

[152] Muhammad Usman. *Supporting Effort Estimation in Agile Software Development*. PhD thesis, Blekinge Institute of Technology, 2015.

[153] Mario E Moreira. Working with story points, velocity, and burndowns. In *Being Agile*, pages 187–194. Springer, 2013.

[154] F Sobiech, B Eilermann, and A Rausch. Using synergies between user stories in scrum. *Lecture Notes on Software Engineering*, 4(2):91, 2016.

[155] Magne Jørgensen. The use of precision of software development effort estimates to communicate uncertainty. In *Software Quality. The Future of Systems-and Software Development*, pages 156–168. Springer, 2016.

[156] Shashank Mouli Satapathy, Aditi Panda, and Santanu Kumar Rath. Story point approach based agile software effort estimation using various svr kernel methods. In *The Twenty-Sixth International Conference on Software Engineering & Knowledge Engineering*, pages 304–307. SEKE, 2014.

[157] Aditi Panda, Shashank Mouli Satapathy, and Santanu Kumar Rath. Empirical validation of neural network models for agile software effort estimation based on story points. *Procedia Computer Science*, 57:772–781, 2015.

[158] Barry W Boehm, Ray Madachy, Bert Steece, et al. *Software cost estimation with Cocomo II with Cdrom*. Prentice Hall PTR, 2000.

[159] Chris Lokan, Terry Wright, Peter R Hill, and Michael Stringer. Organizational benchmarking using the isbsg data repository. *IEEE Software*, 18(5):26, 2001.

[160] Emilia Mendes, Nile Mosley, and Steve Counsell. The need for web engineering: an introduction. In *Web Engineering*, pages 1–27. Springer, 2006.

[161] Burak Turhan, Tim Menzies, Ayşe B Bener, and Justin Di Stefano. On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering*, 14(5):540–578, 2009.

[162] Isabella Wieczorek and Melanie Ruhe. How valuable is company-specific data compared to multi-company data for software cost estimation? In *Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on*, pages 237–246. IEEE, 2002.

[163] Charley Tichenor. A new software metric to complement function points: the software non-functional assessment process (snap). Technical report, DTIC Document, 2013.

[164] Liam O'Brien. A framework for scope, cost and effort estimation for service oriented architecture (soa) projects. In *Software Engineering Conference, 2009. ASWEC'09. Australian*, pages 101–110. IEEE, 2009.

[165] Zheng Li and Jacky Keung. Software cost estimation framework for service-oriented architecture systems using divide-and-conquer approach. In *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*, pages 47–54. IEEE, 2010.

[166] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-oriented programming. In *ECOOP'97Object-oriented programming*, pages 220–242. Springer, 1997.

[167] Gregor J Kiczales, John O Lamping, Cristina V Lopes, James J Hugunin, Erik A Hilsdale, and Chandrasekhar Boyapati. Aspect-oriented programming, October 15 2002. US Patent 6,467,086.

[168] Christian Prehofer. Feature-oriented programming: A fresh look at objects. In *ECOOP'97Object-Oriented Programming*, pages 419–443. Springer, 1997.

[169] Mira Mezini and Klaus Ostermann. Variability management with feature-oriented programming and aspects. *ACM SIGSOFT Software Engineering Notes*, 29(6):127–136, 2004.

Dissemination

Journal Articles

1. **Shashank Mouli Satapathy**, Barada Prasanna Acharya and Santanu Kumar Rath. "Early Stage Software Effort Estimation Using Random Forest Technique Based on Use Case Points.", Vol. 10, Issue 1, pp. 10-17, *IET Software*, IET, 2016. DOI: 10.1049/iet-sen.2014.0122

2. **Shashank Mouli Satapathy**, Mukesh Kumar and Santanu Kumar Rath, "Optimised Class Point Approach for Software Effort Estimation using Adaptive Neuro-Fuzzy Inference System Model," *International Journal of Computer Applications In Technology (IJCAT), Special Issue On: "Current Trends and Improvements In Software Engineering Practices"*, Inderscience, 2016. **(In Press)**

3. **Shashank Mouli Satapathy**, Barada Prasanna Acharya and Santanu Kumar Rath, "Class Point Approach for Software Effort Estimation using Stochastic Gradient Boosting Technique," *ACM SIGSOFT Software Engineering Notes*, Vol. 39, Issue 3, pp. 1-6, ACM, 2014. DOI: 10.1145/2597716.2597725

4. **Shashank Mouli Satapathy**, and Santanu Kumar Rath, "Use Case Point Approach Based Software Effort Estimation using Various Support Vector Regression Kernel Methods," *International Journal of Information Processing (IJIP)*, Vol. 7, No. 4, pp. 87-101, 2013.

5. **Shashank Mouli Satapathy**, Mukesh Kumar and Santanu Kumar Rath, "Fuzzy-Class Point Approach for Software Effort Estimation using Various Adaptive Regression Methods," *CSI Transaction on ICT*, Springer, Vol. 1, Issue 4, pp. 367-380, 2013. DOI: 10.1007/s40012-013-0035-z

Conference Presentations

1. **Shashank Mouli Satapathy**, Aditi Panda and Santanu Kumar Rath, "Story Point Approach based Agile Software Effort Estimation using Various SVR Kernel Methods," *The 26th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 304-308, Vancouver, Canada, 1-3 July 2014.

2. **Shashank Mouli Satapathy** and Santanu Kumar Rath. "Class Point Approach for Software Effort Estimation using Support Vector Regression Technique," *7th India Software Engineering Conference (ISEC)*, pp. 4:1-4:10, (ACM), Chennai, India, 19-21 February 2014. DOI: 10.1145/2590748.2590752

3. **Shashank Mouli Satapathy**, Mukesh Kumar and Santanu Kumar Rath, "Class Point Approach for Software Effort Estimation using Soft Computing Techniques," *The 2nd International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, (IEEE Explorer), pp. 178-183, Mysore, India, 22 - 25 August 2013. DOI: 10.1109/ICACCI.2013.6637167

4. Aditi Panda, **Shashank Mouli Satapathy** and Santanu Kumar Rath, "Neural Network Models for Agile Software Effort Estimation based on Story Points," *International Conference On Advances in Computing, Control and Networking (ACCN)*, IRED, pp. 26-30, Bangkok, Thailand, 21 - 22 February 2015. DOI: 10.15224/ 978-1-63248-038-5-06

5. Aditi Panda, **Shashank Mouli Satapathy** and Santanu Kumar Rath, "Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points," *Third International Conference on Recent Trends in Computing (ICRTC)*, Vol. 57, pp. 772-781, Procedia Computer Science, Elsevier, 12 - 13 March 2015. DOI: 10.1016/j.procs.2015.07.474

Resume

Shashank Mouli Satapathy is presently working as an Assistant Professor at Manipal University Jaipur, Rajasthan, India. He is enrolled in doctoral research program from Department of Computer Science & Engineering at National Institute of Technology (NIT) Rourkela, India. He has completed M. Tech. in Computer Science & Engineering from Kalinga Institute of Industrial Technology (KIIT) University, Bhubaneswar, India in 2012. Prior to that, he has completed his Master of Computer Application (MCA) from Silicon Institute of Technology under Biju Pattanaik University of Technology, India in 2009. His research interests include Software Engineering, Software Effort Estimation, Machine Learning, Program Slicing and Distributed Systems. He is a member of different professional bodies such as IEEE, ACM, IET, CSI. ISCA etc.

E-mail: shashankamouli@gmail.com
Phone: +919438023266

Index